ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO TÉCNICO DE TELECOMUNICACIÓN.
ESPECIALIDAD EN SONIDO E IMAGEN

Título del proyecto:

ESTUDIO Y ELABORACIÓN DE UNA GUÍA DE IMPLEMENTACIÓN DEL ESTÁNDAR IEEE 1451 PARA REDES INALÁMBRICAS

Javier Redondo Expósito

Tutor: Santiago Led Ramos

Pamplona, 1 de Julio de 2010

UNIVERSIDAD PÚBLICA DE NAVARRA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ESTUDIO Y ELABORACIÓN DE UNA GUÍA DE IMPLEMENTACIÓN DEL ESTÁNDAR IEEE 1451 PARA REDES INALÁMBRICAS

Redes de Sensores Inteligentes

Autor: Javier Redondo Expósito
Tutor: Santiago Led Ramos
Fecha: 1 de Julio de 2010

Título:

ESTUDIO Y ELABORACIÓN DE UNA GUÍA DE IMPLEMENTACIÓN DEL ESTÁNDAR IEEE 1451 PARA REDES INALÁMBRICAS

Subtítulo:

Redes de sensores inteligentes

Autor:

D. Javier Redondo Expósito

Tutor:

D. Santiago Led Ramos

Profesor de la Universidad

Tribunal:

Presidente: D

Vocal:

Secretario:

Suplente: D

Calificación:

Pamplona, 1 de Junio de 2010

Resumen:

La necesidad cada vez mayor de medir y controlar parámetros medioambientales, condiciones atmosféricas o realizar una monitorización continua de parámetros fisiológicos en los humanos está creando una gran variedad de sensores inteligentes así como de nuevas tecnologías para comunicar dichos sensores inteligentes y las redes que se crean entre ellos. Por tanto, en la actualidad y como respuesta a esa gran demanda existen una gran diversidad de tecnologías de comunicación y protocolos, tanto por cable (RS232, USB…) como inalámbricas (Bluetooth, Zigbee…). Ahora bien, ante semejante panorama tan diverso de formas y modos de comunicación surgen los problemas de incompatibilidades entre distintos dispositivos, software y protocolos de comunicación perjudicando al usuario final del sistema al obtener poca versatilidad con su sensor o red de sensores.
El estándar IEEE 1451 es creado por el Instituto de Ingenieros Eléctricos y Electrónicos con el fin de solucionar dicho problema y lograr la interoperabilidad de las redes de sensores inteligentes e incluso posibilitar la configuración de sensores "Plug&Play" sin importar la tecnología o forma de comunicación.

Abstract:

The increasing need to measure and monitor environmental parameters, atmospheric conditions or to perform continuous monitoring of physiological parameters in humans is creating a variety of intelligent sensors and new technologies to communicate these smart sensors and networks that are created between them. Therefore, at present and in response to this demand there are a variety of communication technologies and protocols, both wired (RS232, USB ...) and wireless (Bluetooth, Zigbee ...). However, before such a diverse panorama of forms and modes of communication problems arise from incompatibilities between different devices, software and communication protocols hurting the end user of the system to get little versatility with its sensor or sensor network.
IEEE 1451 is created by the Institute of Electrical and Electronics Engineers in order to solve this problem and achieve interoperability of smart sensor networks and sensors even allow "plug & play" regardless of form of communication.

1. DEFINICIONES, ABREVIATURAS Y ACRÓNIMOS

DEFINICIONES

Acknowledge: mensaje de respuesta utilizado para indicar que un comando o mensaje (trigger) ha sido recibido.

Actuator: Un actuador es un transductor que acepta una muestra o muestras de datos y las convierte en una acción. La acción puede ser completamente integrada en el TIM o puede cambiar algo que está fuera del TIM

Address: Un carácter o grupo de caracteres que identifica un registro de un dispositivo o transductor, específica una parte del almacenamiento, o la fuente de datos o del destino.

AddressGroup: una serie de TransducerChannel que mantienen una misma dirección.

Buffer: Una ubicación de almacenamiento de datos intermedios utilizados para compensar la diferencia en la tasa de flujo de datos o el tiempo de ocurrencia de eventos cuando se transmite información de un dispositivo a otro.

Controller: un dispositivo, posiblemente un NCAP, el cual actúa como máster controlando a todos los dispositivos y transductores de la red.

Calibration: El proceso utilizado para determinar la información que debe residir en la Calibration TEDS de apoyo a la corrección de los datos medidos por el transductor.

CommunicationsChannel: A los efectos de este estándar, un CommunicationsChannel se define como una vía de comunicación entre la NCAP y el TIM. Un CommunicationsChannel puede ser utilizado para comunicaciones de datos y otras funciones, así como controlar el funcionamiento de un TIM. Un TIM

tan sólo puede tener un único communication channel (con un NCAP) y el NCAP, por su parte, puede tener varios de ellos (siempre con con TIMs).

Logical CommunicationsChannel: El término "lógica CommunicationsChannel" se utiliza para denotar una CommunicationsChannel cuando las propiedades lógicas son las que son objeto de estudio en lugar de las propiedades físicas. Si el término CommunicationsChannel se utiliza sin un modificador, la lógica CommunicationsChannel es lo que está en estudio.

Physical CommunicationsChannel: el término "physical CommunicationsChannel" es usado para denotar un canal de comunicación cuando las propiedades físicas son las que están en estudio.

Data structure: Un grupo de campos de datos digitales organizados en un orden lógico para algún propósito específico. Una estructura de datos es la plantilla por el cual los datos se almacenan en la memoria del

Digital interface: Un medio de comunicación junto con un protocolo que transfiere información sólo en forma binaria.

Embedded transducer: Un dispositivo que se comporta como un transductor desde el punto de vista del responsable de la red aunque fuera del TIM no se detecte o modifique ninguna magnitud o dato. Los transductores embebidos son útiles para el establecimiento o la lectura de parámetros de funcionamiento de otros transductores.

Event sensor: Un sensor de evento es un sensor que detecta un cambio de estado en el mundo físico. Lo importante es que un cambio de estado se ha producido y / o el instante en el tiempo en el que se ha producido, no el valor del estado o la medida.

Least significant bit (lsb): El LSB es el bit en notación binaria de un número que es el coeficiente de la menor exponente posible.

Meta-TEDS: son TEDS que contienen información sobre un TIM determindao y que es común a todos los TransducerChannel que tiene ese TIM.

Network Capable Application Processor (NCAP): Un NCAP es un dispositivo entre los módulos del transductor (TIM) y la red usuario. El NCAP realiza las comunicaciones de red, comunicaciones TIM, y conversión de datos u otras funciones de procesamiento

Not-a-number (NaN): el dato no es un número

Octet: un grupo de 8 bits

Packet: Un paquete es la información que debe ser transmitida por la capa física entre los dispositivos en una sola.

Sensor: Un sensor es un transductor que convierte una magnitud física, biológica, química o de otro tipo en una señal eléctrica.

Setup time: El tiempo entre la solicitud inicial de una función a realizar, y cuando la tarea se inició en realidad.

Signal conditioning: El acondicionamiento de señales es el procesamiento de la señal del transductor que implica operaciones como la amplificación, la compensación, el filtrado, y la normalización.

Smart actuator: Un actuador inteligente es una versión del actuador de un transductor inteligente.

Smart sensor: Un sensor inteligente es un modelo de sensor inteligente de un transductor.

Smart transducer: Un transductor inteligente es un transductor que proporciona las funciones más allá de las necesarias para generar una representación correcta de una magnitud controlada o medida. Esta funcionalidad simplifica la integración general del transductor a las aplicaciones en un entorno de red.

Transducer: Un transductor es un dispositivo que convierte la energía de una magnitud a otra. El dispositivo podrá ser sensor o actuador.

Transducer Interface Module (TIM): Un módulo que contiene las TEDS, la lógica para implementar la interfaz del transductor, el transductor (o transductores) y la conexión con ellos y cualquier conversión de señal o acondicionamiento de señales.

TransducerChannel: Un TransducerChannel incluye un transductor y todo el acondicionamiento de señales y componentes de conversión asociados a ese transductor.

TransducerChannel address: La concatenación de los alias del TIM y el número del TransducerChannel forman la dirección lógica de un TransducerChannel específico. El alias del TIM es la parte más importante de la dirección TransducerChannel.

TransducerChannel number: Un número de ocho bits asignado por el fabricante a un TransducerChannel de un TIM.

TransducerChannel proxy: Un dispositivo creado para permitir que una colección de TransducerChannels sean tratados como una sola entidad. Un proxy TransducerChannel es similar a un TransducerChannel normal, excepto que no requiere una TEDS TransducerChannel, no puede tener un TEDS calibración, función de transferencia de TEDS o Frequency Response TEDS. Podrá tener otras TEDS. Un proxy TransducerChannel puede responder a los comandos.

Transducer Electronic Data Sheet (TEDS): "hoja de datos electrónicos de un transductor". Una TEDS es un método estandarizado de almacenar datos como la identificación, la calibración, datos de corrección y la información relacionada con el fabricante de un

Trigger: señal o mensaje utilizado para iniciar una acción o mensaje.

Virtual TEDS: una TEDS que se almacena de forma permanente en otro dispositivo que no es el TIM.

INDICE DE CONTENIDOS

INDICE DE ILUSTRACIONES

INDICE DE TABLAS

2. INTRODUCCION

1.1.Antecedentes del proyecto

Los sensores vienen jugando un papel muy importante desde hace unos años hasta ahora. Desde conocer la temperatura de una estancia hasta monitorizar un parámetro fisiológico de una persona, existen un sinfín de sensores que controlan distintas magnitudes para realizar una determinada tarea, de hecho cada día se crean más sensores innovadores para realizar tareas muy específicas en ámbitos tan distintos que pueden ir desde la construcción, automoción incluso seguridad hasta la medicina. Pero no sólo ha habido muchos avances en los sensores en sí, sino en cómo controlarlos, cómo gestionarlos y en cómo interactuar con el dispositivo en definitiva. Para realizar estas funciones y tener un control global sobre ellos se crearon las REDES de SENSORES. El objetivo de estas redes es poder gestionar un gran número de transductores y coordinarlos para que realicen una tarea específica.

La evolución de redes de sensores tiene su origen en iniciativas militares, por esta razón no hay mucha información sobre el origen o necesidad de la idea. La investigación en redes de sensores comenzó cerca de 1980 con el proyecto *Distributed Sensor Networks (DSN)* de la agencia militar de investigación avanzada de los Estados Unidos de América.

Pasando de largo las aplicaciones militares, éstas tienen usos civiles interesantes como vemos a continuación:

- Eficiencia energética: Redes de sensores son utilizadas para controlar el uso eficaz de la electricidad (en Japón está bastante extendido).

- Entornos de alta seguridad: Existen lugares que requieren altos niveles de seguridad para evitar ataques terroristas, tales como centrales nucleares, aeropuertos, edificios gubernamentales de paso restringido. Una red de

sensores podría detectar situaciones que con una simple cámara sería imposible.

- Sensores ambientales: El control ambiental de vastas áreas de bosque o de océano, sería imposible sin las redes de sensores. El control de múltiples variables, como temperatura, humedad, fuego, actividad sísmica así como otras. También ayudan a expertos a diagnosticar o prevenir un problema o urgencia y además minimiza el impacto ambiental del presencia humana.

- Sensores industriales: Dentro de fábricas existen complejos sistemas de control de calidad, el tamaño de estos sensores les permite estar allí donde se requiera.

- Automoción: Las redes de sensores son el complemento ideal a las cámaras de tráfico, ya que pueden informar de la situación del tráfico en ángulos muertos que no cubren las cámaras y también pueden informar a conductores de la situación, en caso de atasco o accidente, con lo que estos tienen capacidad de reacción para tomar rutas alternativas.

- Medicina: Es otro campo bastante prometedor. Con la reducción de tamaño que están sufriendo los nodos sensores, la calidad de vida de pacientes que tengan que tener controlada sus constantes vitales (pulsaciones, presión, nivel de azúcar en sangre, etc.), podrá mejorar sustancialmente.

- Domótica: Su tamaño, economía y velocidad de despliegue, lo hacen una tecnología ideal para domotizar el hogar a un precio asequible.

1.2.Orígenes

Los orígenes de las redes de sensores son una "evolución" de las primeras redes de comunicación, las cuales fueron creadas en Estados Unidos por la agencia DARPA en la década de 1970.

La agencia DARPA (Defense Advanced Research Projects Agency) es una agencia del Departamento de Defensa de los Estados Unidos responsable del desarrollo de nuevas tecnologías para uso militar. Fue creada en 1958 como consecuencia tecnológica de la llamada Guerra Fría, y de la que surgieron, una década después, los fundamentos de ARPANET, red que dio origen a Internet (la "abuela" de Internet).

También durante la Guerra Fría, se creó una de las primeras redes de sensores que se conocen, *Sound Surveillance System* (SOSUS), una red de boyas marinas sumergidas e instaladas en los Estados Unidos para detectar submarinos usando sensores de sonido.

Estandarización de las redes

La no existencia de tecnologías que permitiesen crear redes inalámbricas bajo una norma o protocolo hizo que las primeras soluciones fueran propietarias, tan sólo válidas para la situación en las que hubieran sido implementadas. Poco a poco y conforme se iba avanzando tecnológicamente se fue migrando de estas soluciones particulares hacia soluciones que siguen algún tipo de norma. Las primeras redes *ad hoc* fueron evolucionando hasta llegar hasta las primeras WPANs y WSNs (en el caso de los sensores).

Una *red ad hoc* es una red inalámbrica descentralizada. La red es *ad hoc* porque cada nodo está preparado para reenviar datos a los demás y la decisión sobre qué nodos reenvían los datos se toma de forma dinámica en función de la conectividad de la red. Las redes *ad hoc* son soluciones específicas para un problema específico y por tanto no existe ningún tipo de interoperabilidad debido a que tienen programado un protocolo propietario.

Por otra parte, las redes inalámbricas *Mesh* son aquellas redes en las que se mezclan las dos topologías de las redes inalámbricas, son el siguiente paso lógico a las anteriores *ad hoc*. Básicamente son redes con topología de infraestructura pero que permiten unirse a la red a dispositivos que a pesar de estar fuera del rango de cobertura de los puntos de acceso están dentro del rango de cobertura de alguna

tarjeta de red que directamente o indirectamente está dentro del rango de cobertura de un punto de acceso.

La existencia de múltiples redes inalámbricas (*ad hoc*, sin infraestructura física preestablecida ni administración central) con soluciones específicas y protocolos propietarios evolucionaron, en el caso de los sensores, hacia soluciones (WSN, Wireless Sensor Network) que son un conjunto de elementos autónomos (nodos) interconectados de manera inalámbrica, como se verá un poco más adelante.

Poco a poco se fue migrando a sistemas que ofreciesen mayor interoperabilidad y versatilidad como son las WPAN. Una red inalámbrica de área personal (WPAN) incluye redes inalámbricas de corto alcance que abarcan un área de algunas decenas de metros. Este tipo de red se usa generalmente para conectar dispositivos periféricos o te permite generar una pequeña red de sensores.

La primera gran red de área personal inalámbrica y más conocida es Bluetooth que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz. Esta especificación fue creada por Ericsson en 1994. Se denomina Bluetooth al protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, con una cobertura baja y basada en transceptores de bajo coste.

Posteriormente, finales de 1999, Nokia y Symbol Technologies crearon en 1999 una asociación conocida como WECA (Wireless Ethernet Compatibility Alliance, Alianza de Compatibilidad Ethernet Inalámbrica). Esta asociación pasó a denominarse Wi-Fi Alliance en 2003. El objetivo de la misma fue crear una marca que permitiese fomentar más fácilmente la tecnología inalámbrica y asegurar la compatibilidad de equipos.

De esta forma, en el año 2000, WECA certifica la interoperabilidad de equipos según la norma IEEE 802.11b, bajo la marca Wi-Fi. Esto quiere decir que el usuario tiene la garantía de que todos los equipos que tengan el sello Wi-Fi pueden

trabajar juntos sin problemas, independientemente del fabricante de cada uno de ellos.

1.3.WSN

En los años 90, las redes han revolucionado la forma en la que las personas y las organizaciones intercambian información y coordinan sus actividades. En ésta década seremos testigos de otra revolución; una nueva tecnología permitirá la observación y el control del mundo físico. Los últimos avances tecnológicos han hecho realidad el desarrollo de unos mecanismos distribuidos, diminutos, baratos y de bajo consumo, que, además, son capaces tanto de procesar información localmente como de comunicarse de forma inalámbrica. La disponibilidad de sensores y comunicaciones inalámbricas permitirá desarrollar redes de sensores/actuadores para un amplio rango de aplicaciones. Esto conllevará un necesario desarrollo de modelos físicos, los cuales requieren un análisis y monitorización de datos efectivo y funcional. Un segundo reto a superar es la variabilidad de este nuevo entorno. Mientras un buen sistema distribuido se desarrolla con la fiabilidad como elemento básico, estas nuevas aplicaciones presentan un nivel de aleatoriedad más allá de lo común. Pero la idea dominante radica en las restricciones impuestas por los sistemas en estado inactivo. Estos sistemas deben ser de bajo consumo y larga duración; tanto cuando operan como cuando permanecen a la espera. Además, como en Internet, tenemos sistemas escalables, sin embargo las técnicas tradicionales no son aplicables directamente, así que debemos desarrollar técnicas alternativas.

Cada nodo de la red consta de un dispositivo con microprocesador, sensores y transmisor/receptor, y forma una red con muchos otros nodos, también llamados motas o sensores. Por otra parte, un sensor es capaz de procesar una limitada cantidad de datos. Pero cuando coordinamos la información entre un importante número de nodos, éstos tienen la habilidad de medir un medio físico dado con gran

detalle. Con todo esto, una red de sensores puede ser descrita como un grupo de motas que se coordinan para llevar a cabo una aplicación especifica. Al contrario que las redes tradicionales, las redes de sensores llevarán con más precisión sus tareas dependiendo de lo denso que sea el despliegue y lo coordinadas que estén. En los últimos años, las redes de sensores han estado formadas por un pequeño número de nodos que estaban conectados por cable a una estación central de procesado de datos. Hoy en día, sin embargo, nos centramos más en redes de sensores distribuidas e inalámbricas. Pero, por qué distribuidas e inalámbricas: cuando la localización de un fenómeno físico es desconocida, este modelo permite que los sensores estén mucho más cerca del evento de lo que estaría un único sensor. Además, en muchos casos, se requieren muchos sensores para evitar obstáculos físicos que obstruyan o corten la línea de comunicación. El medio que va a ser monitorizado no tiene una infraestructura, ni para el suministro energético, ni para la comunicación. Por ello, es necesario que los nodos funcionen con pequeñas fuentes de energía y que se comuniquen por medio de canales inalámbricos.

Otro requisito para las redes de sensores será la capacidad de procesamiento distribuido. Esto es necesario porque, siendo la comunicación el principal consumidor de energía, un sistema distribuido significará que algunos sensores necesitarán comunicarse a través de largas distancias, lo que se traducirá en mayor consumo. Por ello, es una buena idea el procesar localmente la mayor cantidad de energía, para minimizar el número de bits transmitidos.

El motivo de haber explicado todo lo anterior es mostrar cuál ha sido el camino y la evolución de las distintas redes (desde de ordenadores hasta las de sensores) existentes hasta llegar a las redes de sensores inalámbricas. Una red de sensores inalámbrica (WSN) es una red inalámbrica que consiste en dispositivos autónomos distribuidos de forma espacial que utilizan sensores para monitorear condiciones físicas y ambientales. Estos dispositivos autónomos, o nodos, se combinan con enrutadores y un gateway para crear un sistema WSN. Los nodos de medida distribuidos se comunican se manera inalámbrica con un gateway central, el cual proporciona una conexión al entorno cableado donde se puede adquirir,

procesar, analizar y presentar sus datos de medida. Para incrementar la distancia y la fiabilidad en una red de sensores inalámbrica, se puede usar enrutadores para lograr un enlace de comunicación adicional entre los nodos finales y el gateway.

Elementos de una WSN típica:

- SENSORES
  - Toman del medio la información y la convierten en señales eléctricas.

- NODOS (Motas)
  - Toman los datos del sensor a y envían la información a la estación base.
  - Estos nodos son pequeñas unidades del tamaño de una caja de cerillas (motas) que tienen solamente unos pocos kilobytes de memoria, un procesador de unos cuantos MHz, una radio de pocos metros de alcance y una o dos pilas (tipo AA, AAA o tipo botón).

- GATEWAY
  - Elementos para la interconexión entre la red de sensores y una red de datos (TCP/IP).

- ESTACIÓN BASE
  - Recolector de datos basado en un ordenador común o sistema embebido.

Las Wireless Sensor Networks, tienen una corta historia, a pesar de ello, ya tenemos a varios fabricantes trabajando en esta tecnología.
CROSSBOW: Especializada en el mundo de los sensores, es una empresa que desarrolla plataformas hardware y software que dan soluciones para las redes de sensores inalámbricas. Entre sus productos encontramos las plataformas Mica, Mica2, Micaz, Mica2dot, telos y telosb.
MOTEIV: Joseph Polastre, antiguo doctorando de un grupo de trabajo de laUniversidad de Berekeley formó la compañía Moteiv. Ha desarrollado la

plataforma Tmote Sky y Tmote Invent. El tipo de mota Tmote Sky será detallado en el siguiente punto, dado que es el que se utiliza en el Instituto de Robótica.
SHOCKFISH: Empresa suiza que desarrolla TinyNode. A partir de este tipo de mota en Laussane han llevado un proyecto semejante al nuestro, en el que implementan una red de sensores en todo el campus de la "Ecole Polytechnique Fédérale deLausanne".

	Micaz	**Mica2**	**Mica2Dot**	**Tmote**	**TinyNode**	**eZ430-RF2480**
Distribuido por	Crossbow			Moteiv	Shockfish	Texas Inst.
Frecuencia reloj	7.37 MHz		4MHz	8 MHz	8 MHz	8 MHz
RAM	4 KB			10 KB	10KB	1 KB
Batería	2 pilas AA		Coin cell	2 pilas AA	Solar	2 pilas AAA
Microprocesador	Atmel Atmega 128L			Texas Instruments MSP430		

Tabla 1:Comparativas de motas

De nuevo, vuelven a coexistir múltiples protocolos como Bluetooth, Zigbee, Wi-Fi… que pueden ser utilizados para crear las redes de sensores inalámbricas, pero el problema que aparece es que en una misma red de sensores no se puede implementar sensores con tecnología Zigbee y tecnología Bluettoth (por ejemplo) y es una vez llegado a este punto donde se comprende la necesidad de implementar un estándar de comunicaciones es una red de sensores que sea capaz de permitir que sensores de distintas tecnologías puedan trabajar juntos para realizar su tarea.

Con la llegada de las redes de sensores inteligentes y la gran demanda de estos en diversos ámbitos y usos, se abrió un amplio mercado donde proliferaron diversas tecnologías y aplicaciones.

1.4.ISO/IEEE 11073

Un ejemplo muy claro de esto es el trabajo que se lleva realizando desde finales de los 80s en el campo de la medicina, un estándar de comunicaciones para dispositivos médicos, ISO/IEEE11073.

Se trata de una familia de normas, promovidas por el *Institute of Electrical and Electronics Engineers* (IEEE), consorcio de fabricantes e instituciones, y adoptadas como estándar internacional de ISO (ISO11073), y que se basa en varios trabajos del CEN. Agrupa diversas normas CEN e IEEE anteriores para cubrir los diferentes niveles del modelo OSI: MIB para los niveles OSI inferiores, e INTERMED y VITAL para los niveles superiores.

La familia de normas X73 son un conjunto de normas para conectividad completa de dispositivos médicos, empezando en el nivel físico (ya sea cable o inalámbrico) hasta la representación abstracta de información y los servicios usados para su gestión e intercambio. El resultado es un conjunto único de normas desarrolladas y adoptadas por todos los países, que aporta interoperabilidad, transparencia, funcionalidad *Plug&Play*, y facilidad de uso y configuración. Algunos de los fundamentos básicos de X73 recuperan los conceptos clásicos de comunicación basados en el modelo genérico OSI transmisión cliente/servidor; intercambio de primitivas de servicio, tramas de protocolo de datos (*Protocol Data Unit*, PDU), etc.

La familia de normas ISO/IEEE11073 (X73) ha sufrido un proceso evolutivo desde el comienzo de su desarrollo hasta la actualidad, en el que multitud de ingenieros han colaborado en paralelo con universidades, instituciones y entidades internacionales. Su primera versión X73-PoC, trataba la problemática de la interoperabilidad en la comunicación entre los diferentes dispositivos médicos (*Medical Devices*, MDs) en torno al paciente (en el Punto de Cuidado, PoC) y un elemento concentrador (denominado *gateway*) que centraliza los datos médicos adquiridos de cada uno de ellos y estandariza el formato de dicha información para su posterior envío hacia el hospital o el servidor de telemonitorización propio del servicio de e-Salud. El posterior desarrollo de nuevos MDs *wearables*, con sensores

de alta calidad y sobre tecnologías *wireless* (como Bluetooth o ZigBee), y el incremento de accesos de banda ancha a redes multimedia, aceleró la evolución del estándar X73 hacia una versión optimizada y adecuada a estas nuevas tecnologías y contextos ubicuos: X73-PHD.

En términos generales, X73-PHD posibilita la conexión entre MDs y sistemas externos (*Compute Engines*, CE) como extensión del denominado *gateway* en X73-PoC. Esto hace posible que se evolucione desde los casos de uso más conocidos y trabajados en soluciones de e-Salud (telemonitorización domiciliaria, seguimiento de ancianos o pacientes crónicos, etc.), hacia un nuevo contexto ubicuo de soluciones de u-Salud (proporcionado por X73- PHD) que abren el abanico de nuevos casos de uso a los que X73 debe dar respuesta: entrenadores personales, medicina deportiva, atención personalizada de pacientes, escenarios móviles con dispositivos *wireless*, etc.

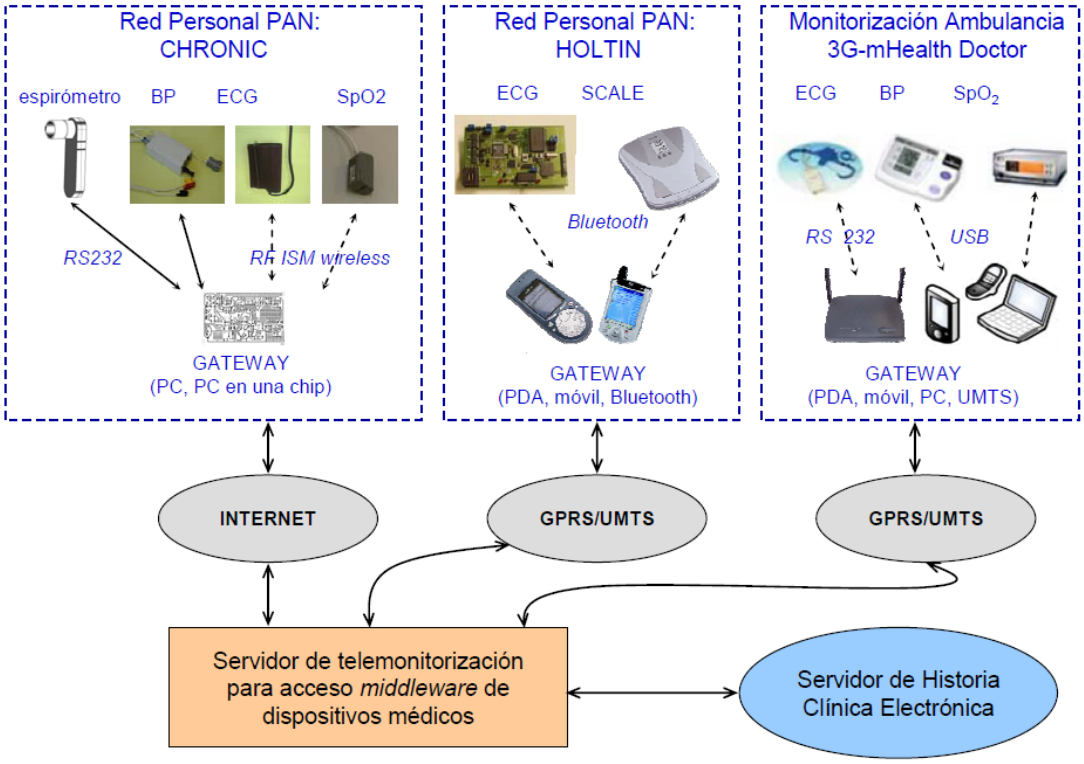


Ilustración 1: Ejemplo migración de PoC a PHD en IEEE 11073

El sector industrial ha hecho esfuerzos por tener en claro la definición, funcionalidad y estándares de comunicación en sensores inteligentes. La meta es poder lograr la interoperabilidad en un amplio rango de aplicaciones. Un sensor inteligente con su adecuada capacidad deberá poder actuar como un sensor aislado, relacionarse punto a punto con otros sensores o actuadores o actuar como un nodo inteligente dentro de la red. La interoperabilidad será la implementación de varias interfaces a través de diferentes redes para operación de redes independientes.

Bajo estas consideraciones el desarrollo de este tipo de productos se ha proliferado y por lo tanto diversidad de tecnología. Por eso desde 1993, la IEEE y el *National Institute for Standards and Testing* (NIST) iniciaron una actividad que ha conducido al desarrollo de un estándar para sensores inteligentes, que le han asignado el número de IEEE-1451. El objetivo es desarrollar una familia de estándares para interconectar sensores inteligentes (NIST-1451, 2008). Este conjunto de estándares está encaminado a facilitar el desarrollo de dispositivos inteligentes, que puedan interconectarse en redes, sistemas e instrumentos, los cuales puedan incorporarse a sensores y redes existentes y emergentes.

Básicamente existen dos clases de dispositivos descritos por el estándar: el procesador de las aplicaciones en la red (NCAP, *Network Capable Applications Processor*), y el módulo interfaz de los transductores (TIM). El NCAP forma el puente entre la red y el modulo de los transductores.

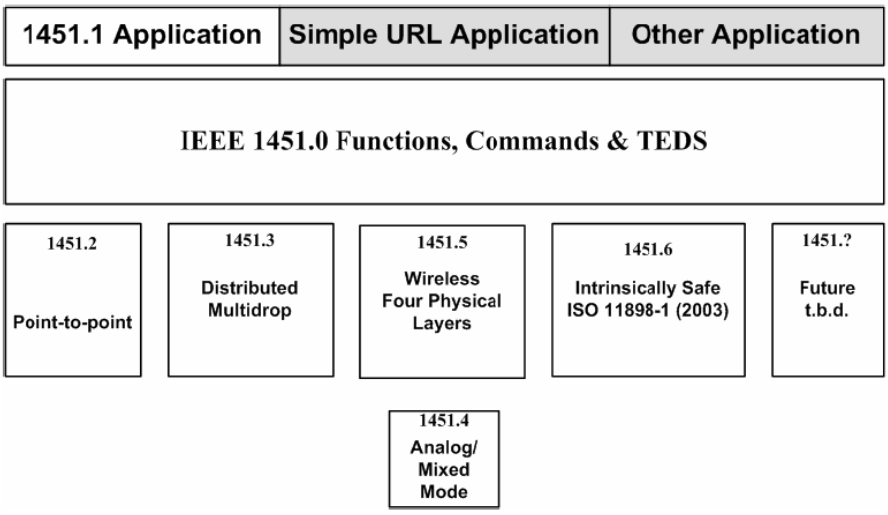


Ilustración 2: Estructura

3. OBJETIVOS Y MOTIVACIÓN

3.1.Motivación

El actual desarrollo de los sistemas integrados para la monitorización/supervisión de todo tipo de escenarios y casos de uso (Salud Personal, Vida Independiente, Bosques, Confort en Edificios Públicos y Privados, etc.) basándose en la utilización de Tecnologías Inalámbricas como Bluetooth, Zibgee, 6LowPan, etc., provoca la aparición de sistemas comerciales propietarios, los cuales no interaccionan entre ellos resultando costoso, tanto en tiempo como en presupuesto, la inclusión de nuevos sensores de otros fabricantes.

La mejor forma de evitar estas limitaciones, y de paso poder aumentar el potencial número de clientes, se basa en la utilización de estándares que nos fijen tanto el almacenamiento de los datos como el protocolo de comunicación independientemente de la tecnología de transporte utilizada. En este sentido, una de las propuestas más prometedoras para casos de uso de Redes Sensores Inalámbricos (Wireless Sensor Networks) es IEEE 1451.

Cuando se tiene un sistema o una red de sensores con un tamaño de dispositivos considerable y además dicha red la conforman sensores que operan a través de distinta tecnología o protocolo de comunicaciones, es muy necesario el tener un estándar de comunicación para hacer que se "entiendan" los distintos sensores entre sí. Es un punto crítico, puesto que hay que encontrar la solución de compromiso ideal entre el coste (en cuanto a necesidades hardware y computacional) de la implementación y la versatilidad necesaria para la aplicación.

3.2.Objetivos

Este proyecto pretende ayudar a valorar y conocer las posibilidades que aporta la implementación del estándar IEEE 1451 en una red de sensores inteligentes junto con todas las implicaciones y limitaciones que supone. Para ello se ha realizado un estudio minucioso del estándar de sensores inteligentes para posteriormente poder comprobar la viabilidad del uso de este estándar frente al uso de protocolos propietarios que aunque inicialmente puedan resultar más sencillos de llevar a cabo a la práctica (tanto por características del dispositivo como de código) luego pueden presentar incompatibilidades a la hora de realizar ampliaciones o modificaciones en la aplicación.

Como se ha explicado anteriormente en la actualidad existen múltiples formas y protocolos de comunicación para implementar redes de sensores inteligentes, debido descenso del coste y a la oferta/demanda, cada día hay más variedad tanto de dispositivos como de tecnologías, por lo que parecería inexcusable el no valorar ni tan sólo un instante la posibilidad de implementar el estándar IEEE 1451 (al menos una parte de él). Es un punto crítico en el que se debe valorar muy cuidadosamente qué es lo que va a aportar el estándar a la aplicación, es decir, que grado de versatilidad puede adquirir dicha aplicación y entender que coste y recursos van a suponer. También es muy importante valorar el futuro de la red de sensores, es decir, si es probable que se amplíe o se modifiquen aspectos de la aplicación a corto plazo…

Para resolver algunos de los problemas anteriormente descritos pueden adoptarse diversas soluciones, algunas de las cuales nosotros hemos decidido no contemplar por el elevado consumo de recursos y la poca sostenibilidad del servicio que conllevan.

El análisis de la problemática y los requisitos que se nos plantean, nos permite llegar a la especificación de los objetivos finales del proyecto, que son los que se indican a continuación:

- Realizar estudio profundo IEEE 1451
- Comprender los mensajes que se intercambiar distintos componentes del estándar
- Realizar una guía de implementación de IEEE 1451
- Proponer y sugerir aplicaciones del estándar y una implementación en eZ430-RF2480

4. LOS SENSORES INTELIGENTES Y LAS REDES

El mercado de sensores es muy diverso. Los sensores se utilizan en la mayoría de las industrias, por ejemplo, control aeroespacial, automoción, biomedicina, construcción, control industrial, la fabricación... Los fabricantes de sensores están buscando formas de construir a bajo coste los sensores inteligentes para satisfacer la continua demanda de aplicaciones más sofisticadas junto con facilidad de uso. Las redes se están convirtiendo en elementos omnipresentes en todos los ámbitos en los países industrializados y está causando un cambio clave en el negocio de los sensores.

El rápido desarrollo y aparición de sensores inteligentes y tecnologías de red han hecho que la creación de redes de transductores inteligentes (sensores y actuadores) una solución muy económica y atractiva para una amplia gama de aplicaciones de medición y control.

Sin embargo, con la multitud de especificaciones diferentes de redes y de las incompatibles surgidas, han creado un cierto grado de confusión e incertidumbre. Está claro que deben de existir diferentes redes para resolver problemas específicos y distintos. Sin embargo, parece que la industria está en una encrucijada y esta situación ha impuesto un coste económico innecesario tanto para los usuarios finales como a los proveedores de transductores. El sector industrial ha hecho esfuerzos por tener en claro la definición, funcionalidad y estándares de comunicación en sensores inteligentes. La meta es poder lograr la interoperabilidad en un amplio rango de aplicaciones. Un sensor inteligente con su adecuada capacidad deberá poder actuar como un sensor aislado, relacionarse punto a punto con otros sensores o actuadores o actuar como un nodo inteligente dentro de la red (Randy 2000). La interoperabilidad será la implementación de varias interfaces a través de diferentes redes para operación de redes independientes.

En vista de esta situación, el Comité Técnico de Tecnología de Sensores del Instituto de Ingeniero Eléctricos y Electrónicos *(IEEE) 's Instrumentación y Medición de la Sociedad* ha patrocinado una serie de proyectos, designados como IEEE P1451, para abordar estas cuestiones a través del desarrollo de una familia de normas de interfaces transductores inteligentes para conectar a las redes de transductores. Cuando estas interfaces estandarizadas están en su lugar, los fabricantes de sensores pueden diseñar sus dispositivos a un solo conjunto de especificaciones de los transductores y la conectividad de redes. Se pretende que esto alivie la incertidumbre y permita el rápido desarrollo de sensores y actuadores inteligentes para su uso en las redes. A la larga, conducirá muy probablemente a un coste de desarrollo más bajo para los fabricantes de sensores inteligentes y una proliferación de sensores inteligentes en el mercado.

Por lo tanto, la aparición de sensores inteligentes y de las redes pondrá a disposición una amplia variedad de productos para los usuarios a elegir de acuerdo a sus necesidades. Si esta tendencia continúa, con el tiempo bajará el coste total del sistema y se podrá ampliar el dominio de aplicación para aplicaciones de control distribuido para los usuarios

4.1.Sensores inteligentes

Un sensor no es más que un transductor el cual convierte una magnitud física (temperatura, fuerza, luminosidad…) en una magnitud eléctrica (voltaje, corriente, impedancia). Esta magnitud eléctrica, es una señal analógica y lo que es más importante, necesita un interfaz con el sistema para ver cuál es la medida que nos interesa. En la mayoría de ocasiones estos interfaces son específicos o puede ser que el sensor utilice una tarjeta de adquisición, es decir, que los sensores son poco versátiles.

En contra de lo dicho anteriormente, los Sensores Inteligentes (*Smart Sensors*), son dispositivos dan la información de una magnitud física en un formato

adaptado al sistema de medida. Un sensor inteligente es aquel que combina la función de detección y alguna de las funciones de procesamiento de la señal y comunicación. Dado que estas funciones adicionales suele realizarlas un microprocesador, cualquier combinación de sensor y microprocesador se denomina sensor inteligente. Aunque no tiene que ser un elemento monolítico, se sobre entiende que un sensor inteligente está basado, total o parcialmente, en elementos miniaturizados, y con un encapsulado común.

Un sensor inteligente es inevitablemente más caro que un sensor convencional. Pero si además del coste de compra se consideran el de mantenimiento, fiabilidad, etc., el costo total de un sensor convencional puede ser mucho mayor. El nivel de complejidad de un sensor inteligente puede ser muy variado.

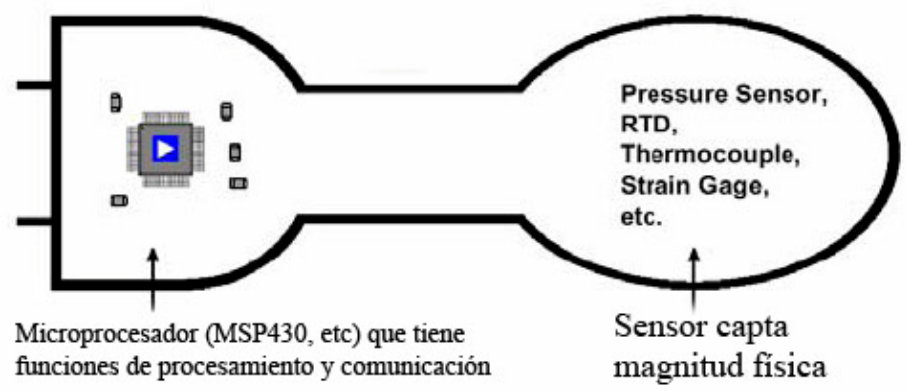
Información típica que puede ofrecer uno de estos sensores:

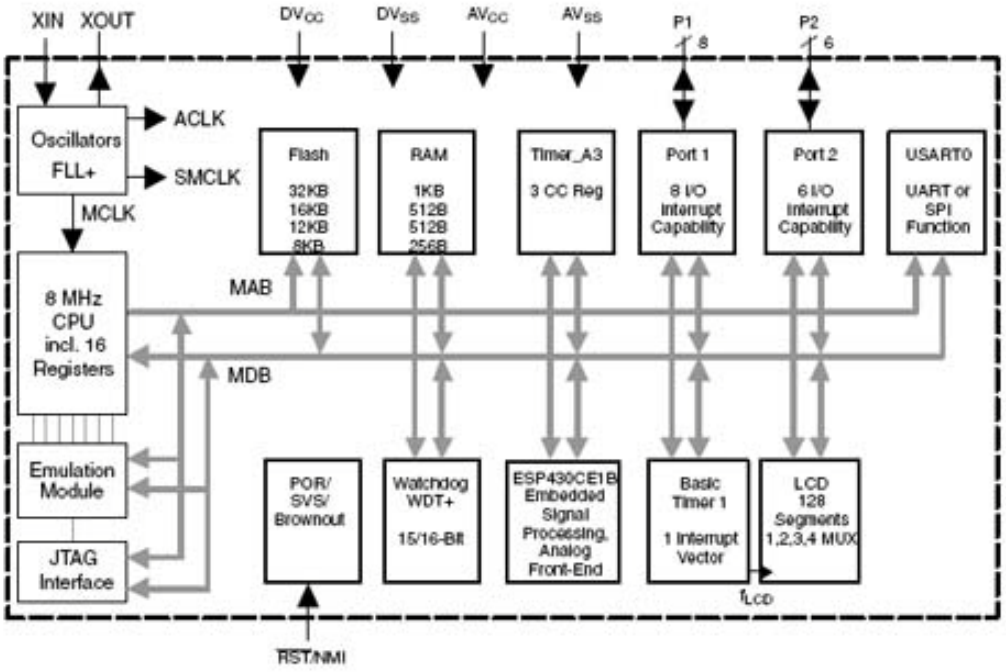
- Datos de la magnitud medida
- Información de calibración
- Información de cómo interpretar los datos proporcionados

Combinan la detección de una magnitud con funciones de procesamiento de señal y comunicación. Una vez introducido el concepto de "inteligencia" en los sensores, existen distintos grados de inteligencia dependiendo de las diversas funciones que lleve integradas como por ejemplo:

- Adaptación y acondicionamiento.
- Conversión A/D
- Comunicaciones/Transmisión de la información

Dichas funciones están integradas en un microprocesador, el cual está en comunicación con el sensor. Por ejemplo, el microprocesador de Texas Instruments MSP430.





MSP430

Además, son dispositivos de dimensiones reducidas, capaces de tomar varias magnitudes de medida simultáneamente y con gran funcionalidad.

4.2.Redes de Sensores

Una red de sensores puede ser descrita como un grupo de "sensores inteligentes" (también llamados motes o motas) que se coordinan para llevar a cabo una aplicación específica.

La evolución de redes de sensores tiene su origen en iniciativas militares, por eso no hay mucha información sobre el origen de la idea. La investigación en redes de sensores cerca de 1980 comenzó con el proyecto *Distributed Sensor Networks (DSN)* de la agencia militar de investigación avanzada de Estados Unidos.

Al contrario que las redes tradicionales, las redes de sensores llevan con más precisión sus tareas dependiendo de lo denso que sea el despliegue y lo coordinadas que estén. En los últimos años, las redes de sensores han estado formadas por un pequeño número de nodos que estaban conectados por cable a una estación central de procesado de datos. Hoy en día, sin embargo, nos centramos más en redes de sensores distribuidas e inalámbricas. Pero, por qué distribuidas e inalámbricas: cuando la localización de un fenómeno físico es desconocida, este modelo permite que los sensores estén mucho más cerca del evento de lo que estaría un único sensor.

Ahora, ¿cómo se comunican estas redes?, ¿qué protocolo o estándar utilizan?, ¿qué posibilidades me ofrece cada uno de ellos? Es aquí donde aparece IEEE 1451.

5. IEEE 1451. HACIA LA CONVERGENCIA DE LAS REDES DE SENSORES INTELIGENTES

En la actualidad existen multitud de redes de sensores inteligentes cuyos datos pueden ser consultados a través de Internet, pero el problema radica en que cada red utiliza sus propios estándares, protocolos y formatos de representación de datos.

Las características deseables de los sensores de la red son:

- Fácil instalación
- Auto-identificación
- Auto-diagnóstico
- Confiabilidad
- Coordinación con otros nodos
- Funciones software y de tratamiento digital de la señal
- Protocolos de control y de interfaz de red estándares

IEEE 1451 es el estándar para sensores inteligentes (smart sensors) creado por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE), el cual es una asociación sin ánimo de lucro formada por más de 360.000 miembros repartidos por todo el mundo.

5.1.Inicios del estándar IEEE 1451

En Septiembre de 1993, el Instituto Nacional de Estándares y Tecnología (NIST) y el comité del Instituto de Ingenieros Electrónicos y Eléctricos (IEEE), se reunieron para tratar sobre los interfaces en las comunicaciones de sensores inteligentes y la posibilidad de crear un interfaz estándar. Se llegó al acuerdo de crear una interfaz común de comunicación para transductores inteligentes. Desde entonces, se han formado distintos grupos de trabajo para abordar diferentes aspectos de la norma del interfaz.

El grupo de trabajo P1451.1 tiene por objeto definir un modelo de objetos común para transductores inteligentes, junto con especificaciones de interfaz para los componentes del modelo.

El grupo de trabajo P1451.2 tiene por objeto la definición de un módulo de interfaz inteligente transductor (STIM), un transductor de hoja de datos electrónicos (TEDS), y una interfaz digital para acceder a los datos.

Por su parte, el objetivo del grupo P1451.3 trata definir una interfaz de comunicación digital para los sistemas de distribución multipunto (Sistemas Multidrop).

El objetivo de P1451.4 es la definición de una mezcla de modo de protocolo de comunicación para transductores inteligentes. Esta familia de estándares IEEE 1451 está diseñada para trabajar de forma sincronizada para facilitar la conexión de sensores y actuadores en un dispositivo o en una red.

Los grupos de trabajo crearon el concepto de sensores inteligentes para controlar la interoperabilidad de redes. Este concepto de sensor / interoperabilidad de la red se han demostrado en dos congresos sobre sensores en Boston y Chicago.

El propósito subyacente de esta familia de normas es permitir a los fabricantes que los elementos construidos de un sistema sean interoperables. Para lograr este objetivo, la familia IEEE 1451 divide las partes de un sistema en dos

categorías de dispositivos. Una de ellas es la red de procesador de aplicaciones capaces (NCAP), que funciona como una puerta de enlace entre la red de los usuarios y los módulos de interfaz del transductor (TIM). El NCAP es un dispositivo que tiene dos interfaces. La interfaz física para los usuarios de la red no se especifica en cualquiera de esta familia de normas. IEEE Std 1451.1 proporciona un modelo de objetos lógicos para esta interfaz. Otras aplicaciones también pueden ser utilizadas a elección del fabricante. La interfaz de comunicaciones entre el NCAP y TIMs se especifica en los demás miembros de IEEE 1451 (como 1451.5…).

Los distintos fabricantes pueden construir NCAPs y TIMs, y si ambos cumplen con IEEE 1451, serán interoperables.

5.2.Miembros pertenecientes a IEEE 1451

IEEE-1451.1, aprobado en 1996, define un modelo de objeto del software común para el NCAP, que tiene distintas versiones dependiendo de la red y el tipo de módulo de los transductores (Artaud et al, 2004 y NIST-1451, 2008).

IEEE-1451.2 aprobado en 1997, en el cual se define la forma en que los transductores son incluidos en el módulo interfaz del transductor inteligente, STIM por sus siglas: *Smart Transducer Interface Module*, conectado al NCAP por medio de una conexión punto a punto (P2P). Y también define el formato electrónico de las especificaciones del transductor TEDS, por sus siglas *Transducer Electronics Data Sheet* (TEDS) (Artaud et al, 2004 y NIST-1451, 2008).

IEEE-1451.3 aprobado en el 2003**,** define la forma en que los transductores son incluidos en el módulo interfaz del transductor inteligente, STIM por sus siglas: *Smart Transducer Interface Module*, conectado al NCAP por medio de una conexión multipunto. Y también define el formato electrónico de las especificaciones del transductor TEDS, por sus siglas *Transducer Electronics Data Sheet* (TEDS) (Artaud etal, 2004 y NIST-1451, 2008).

IEEE-1451.4 aprobado en el 2004, define la forma en que se agrega la función de auto identificación a sensores y actuadores analógicos. Define el concepto de un transductor de modo mixto que alimenta una interfaz analógica y digital. Y también define el formato electrónico de las especificaciones del transductor TEDS (Artaud et al, 2004 y NIST-1451, 2008).

IEEE 1451.0 aprobado en el 2007, contiene un nuevo estándar que define un conjunto de funciones comunes, protocolos de comunicación y formatos TEDS para varias formas de comunicación. Con el propósito de lograr la interoperabilidad del estándar y simplificar su aplicación.

Es muy importante señalar que la estructura de las TEDS especificadas en esta parte del estándar varía con respecto a la que se define tanto en IEEE 1451.2, **IEEE 1451.3** y **IEEE 1451.4**. No obstante sí que tienen un punto para entenderse unas con otras por parte de las APIs.

IEEE-1451.5 aprobado en el 2007, define o establece el estándar para una comunicación inalámbrica para tecnologías Wi-Fi, bluetooth, Zigbee y 6LowPAN, y el formato de la información para el transductor. También define el formato electrónico de las especificaciones del transductor TEDS, de acuerdo al formato IEEE-1451 (NIST-1451, 2008 y IEEE P1451.5 Project, 2008).

IEEE-P1451.6 actualmente se encuentra en desarrollo, el propósito de este estándar es definir la implementación de los conceptos del TEDS de sobre una red con CAN. El propósito es desarrollar un puerto de entrada simple para una red de sensores en cascada combinando las especificaciones del IEEE 1451 y el CAN (NIST-1451, 2008 y IEEE P1451.5 Project, 2008).

También es conveniente mencionar que ya se está trabajando en el proyecto del estándar **IEEE-P1451.7**, se trabaja desde el 2007 y está previsto desarrollarlo hasta el 2011, el propósito de este estándar es proveer métodos e interfaces para interconectar las tarjetas de radio identificación, *Radio Frequency IDentification* (RFID) y transferencia de información entre la infraestructura RFID. También se trabajará en la definición de dispositivos y equipos para la interoperabilidad.

De todos los miembros de IEEE 1451 que se acaban de describir brevemente, son IEEE 1451.0 y IEEE 1451.5 los miembros en los que se centra este proyecto. Se describirán sus funcionalidades y características más a fondo.

6. IEEE 1451.0

6.1.Introducción

El estándar IEEE 1451.0 define un conjunto común de comandos, formatos TEDS, y protocolos de comunicación para la familia de estándares IEEE 1451, para que contribuyan a conseguir interoperabilidad, a nivel de datos, entre los distintos miembros de IEEE 1451. Las aplicaciones de IEEE 1451se comunican con los servicios de IEEE 1451.0 a través de la interfaz IEEE 1451.0 (Transducer Service Interface) del NCAP. El módulo de comunicaciones de IEEE 1451.X del NCAP puede comunicarse con el módulo de comunicaciones IEEE 1451.X del TIM para recibir datos del transductor y TEDS enviadas a través de IEEE 1451.X, como IEEE 1451.2-1997, IEEE 1451.3-2003, IEEE 1451.5-2007, IEEE p1451.6 y el IEEE p1451.7. IEEE 1451.0 proporciona una interfaz unificada para el estándar de la familia IEEE 1451 de transductores inteligentes. Para que los transductores inteligentes puedan alcanzar funcionalidad *Plug&Play*, el NCAP y el TIM deberán ajustarse a la norma IEEE 1451.0.

Las implementaciones del NCAP y del TIM deben ser compatibles con las especificaciones requeridas de las funciones, estructuras de mensaje, los comandos, las TEDS, y el respectivo módulo de comunicaciones de IEEE 1451.X.

Al tratarse de un estándar que se lleva construyendo desde hace unos quince años, los nuevos miembros pueden originar en ocasiones incompatibilidades con otros más antiguos. Por ejemplo, cuando en el año 2007 se creó el miembro IEEE 1451.0, no era compatible con IEEE 1451.2-1997, pero el grupo de trabajo IEEE p1451.2 revisó el estándar para hacerlo compatible con el estándar IEEE 1451.0. No obstante, y como se explicará más adelante, ya se ha encontrado la forma de hacer compatible las TEDS de 1451.0 con las de 1451.2. Es aquí donde radica la importancia de que se trate de un estándar abierto.

En la siguiente figura se muestra el stack (arquitectura) de IEEE 1451. Como se observa el miembro principal a día de hoy para realizar cualquier implementación es IEEE 1451.0 puesto que es el responsable de controlar y permitir que el resto de miembros puedan trabajar juntos.

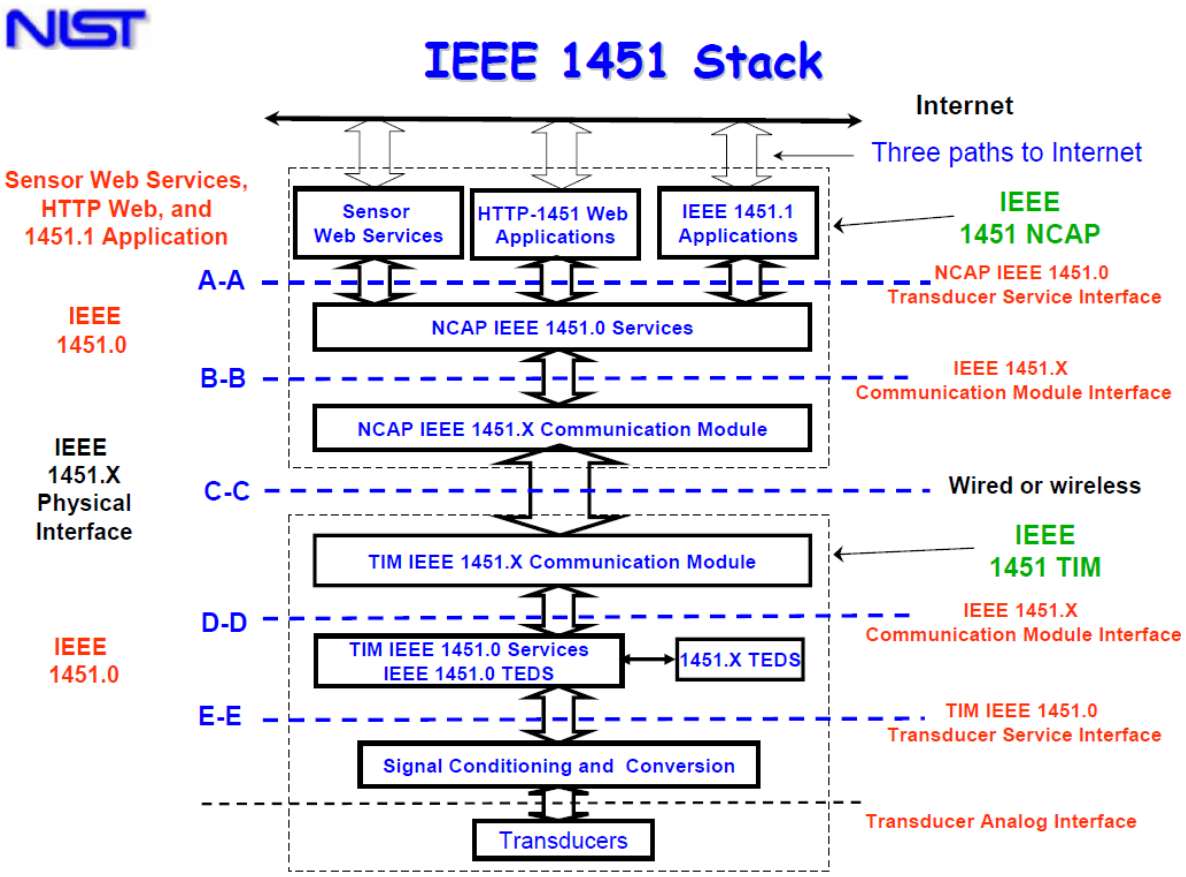


Ilustración 3: Arquitectura IEEE 1451

6.2.Estructuras de los Mensajes

Estos mensajes que se presentan a continuación, son enviados a través del Módulo de Comunicaciones, se generan en la capa IEEE 1451.0 del dispositivo NCAP (y de forma opcional en la capa IEEE 1451.0 del dispositivo TIM, si se quiere que éste inicie la comunicación). Además hay que tener en cuenta que dicha

estructura es conceptual y que dichos mensajes pueden sufrir modificaciones en la capa física del módulo de comunicaciones.

La información se envía a través de octetos (8 bits).

1 Octeto
Destinatario mensaje- Transducer Channel (octeto más significativo)
Destinatario mensaje- Transducer Channel (octeto más significativo)
Clase de Comando
Función del Comando
Longitud (octeto más significativo)
Longitud (octeto menos significativo)
Octetos dependientes del Comando
…
…

Tabla 1: Estructura de los mensajes

El destinatario del mensaje, Transducer Channel Number, está formado por 16 bits y puede tomar valores entre 1 y 0x7FFF.

La clase de Comando especifica que familia pertenece el comando

La Función del Comando especifica la función dentro de la clase de comando.

La Longitud especifica el número de octetos que tiene la parte de Octetos dependiente del Comando.

Octetos dependientes del Comando contiene la información de la Función Comando.

Mensajes Respuesta

Los Reply Messages son enviados para contestar peticiones de distintos comandos.

1 Octeto
Éxito/ Error- Success/FailFlag
Longitud (octeto más significativo)
Longitud (octeto menos significativo)
Octetos dependientes de la respuesta al Comando
…
…

Tabla 2: Estructura mensaje respuesta

Si el valor del octeto *Succes/FailFlag* es distinto de cero, se indica que el comando se ha completado satisfactoriamente. Si es cero, da error, y se manda al origen un mensaje de error para tratar el motivo del mismo.

La *Longitud* especifica el número de octetos que tiene la parte de Octetos dependientes de la respuesta al Comando.

Por su parte, los *Octetos dependientes de la respuesta al Comando* contienen la respuesta al comando que previamente le han pedido, dichos octetos contienen los parámetros que son definidos dentro del estándar en la sección *Commands*.

A continuación se muestra en la siguiente figura lo anteriormente explicado.

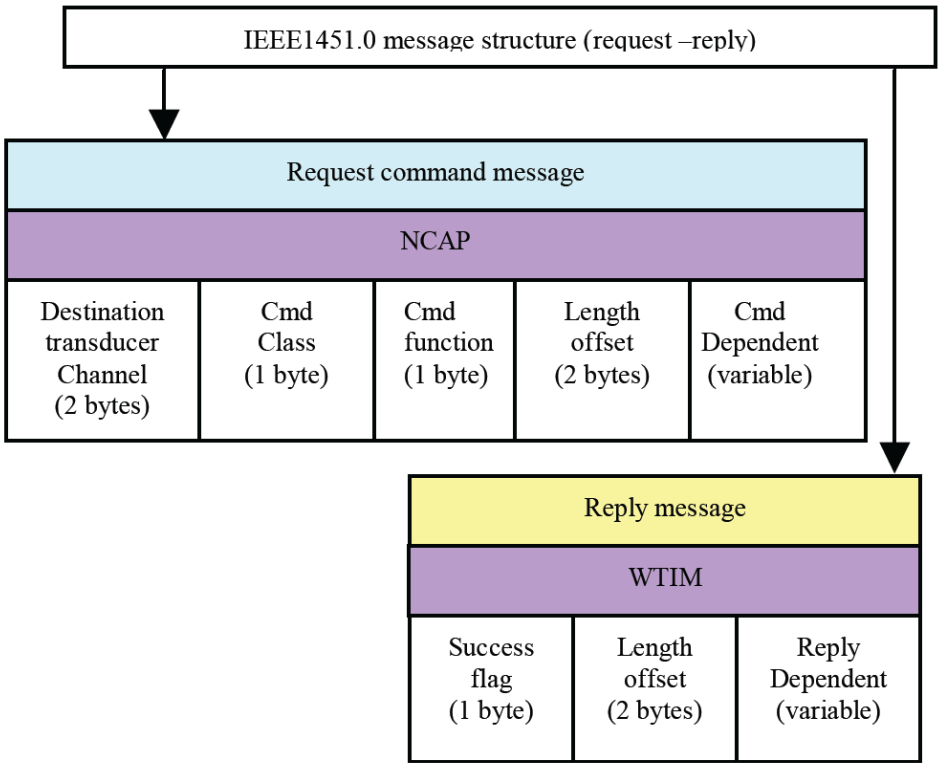


Ilustración 4: Capa en la que se originan los mensajes.

Por otra parte, cuando es el propio TIM el que inicia la comunicación, la estructura del mensaje a enviar es como se muestra:

1 Octeto
Emisor mensaje- Transducer Channel (octeto más significativo)
Emisor mensaje- Transducer Channel (octeto más significativo)
Clase de Comando
Función del Comando
Longitud (octeto más significativo)
Longitud (octeto menos significativo)
Octetos dependientes del Comando
…
…

Tabla 3: Estructura mensaje cuando lo crea un TIM.

Como se puede observar es exactamente la misma estructura que cuando el NCAP inicia la comunicación con la salvedad de que en vez de enviar la dirección del destino, se envía la del TransducerChannel de origen.

Hay que tener en cuenta, y como se explicará más adelante (concretamente en el apartado del Módulo de Comunicaciones), que no es obligatorio que el TIM inicie la comunicación dependiendo de cómo configuremos el Módulo de Comunicaciones.

6.3.Comandos

Principalmente los comandos se dividen en dos grandes grupos: *Standards Commands* y *Manufactured-defined Commands*.

Los *Manufactured-defined Commands* no se especifican en IEEE 1451.0, siendo el fabricante el que decide si desea o no especificar nuevos comandos con el fin de obtener nuevos servicios, este punto es muy importante puesto que permite definir al fabricante una serie de comandos permitiendo que no salga fuera del

estándar y a la vez permitiendo al fabricante dar unos servicios específicos pudiendo diferenciarse de otros.

Los *Standard Commands*, sí se encuentran definidos en IEEE 1451.0, permitiendo ser enviados al TIM o TransducerChannel para que realice una operación determinada. A continuación se muestran las principales familias de comandos, que se corresponden con la Clase de Comando.

cmdClassId	Nombre del Atributo	Categoría
0	Reserved	Reservado
1	CommonCmd	Comandos Comunes al TIM y TransducerChannel
2	XdcrIdle	Estado en reposo del transductor
3	XdcrOperate	Estado en funcionamiento del transductor
4	XdcrEither	Estado en reposo o funcionamiento
5	TIMsleep	Estado dormido (bajo consumo)
6	TIMActive	Comando del TIM activo
7	AnyState	Cualquier estado
8-127	ReservedClass	Reservado
128-255	ClassN	Abierto a fabricantes

Tabla 4: Clases se comandos.

Un TIM puede generar una respuesta a un comando en virtud de cualquiera de las dos siguientes circunstancias. La primera, es cuando un comando que le han enviado requiera una respuesta. Un ejemplo de esta situación es un comando de *QUERY TEDS*. La segunda circunstancia es cuando está activado el protocolo sobre el *Status-Event*. En este caso, el TransducerChannel o TIM transmite cada vez que haya un no enmascarado (non-masked) cambiando el *Status-Event*.

Tras haber estudiado detenidamente todos estos comandos y observando también algún ejemplo encontrado en la red, he concluido que si queremos tener un sistema que sea compatible (o se le pueda reconocer) con IEEE 1451.0 debe tener implementado (como mínimo) el comando *CommonCommands*. Dicho comando contiene funciones que controlan y gestionan las TEDS. Obviamente, contra más comando implementemos en nuestro sistema más fiel será al estándar.

Dentro la *ClassCommand* (clase de comando) se define el *FunctionCommand* (función del comando), por ejemplo, dentro de la clase CommonCommand se encuentra la función QueryTEDS, la cual hace operaciones con las TEDS, y dentro de dicha función se especifica la operación que lleva a cado así como los atributos que utiliza y lleva a cabo.

Cmd Function Id	Commando	Estado		Respuesta	Requerido/ Opcional
		TransducerChannel	TIM		
0	Reservado	-	-	-	-
1	Query TEDS	Cualquiera	Activo	Sí	Requerido
2	Leer segmento TEDS	Cualquiera	Activo	Sí	Requerido
3	Escribir segmento TEDS	Cualquiera	Activo	No	Requerido
4	Actualizar TEDS	Cualquiera	Activo	Sí	Requerido
5	Run self-test	Reposo	Activo	No	Requerido
6	Escribir servicios mask	Cualquiera	Activo	No	Requerido
7	Leer servicios mask	Cualquiera	Activo	Sí	Requerido
8	Leer registro Status-Event	Cualquiera	Activo	Sí	Requerido
9	Leer registro Status-Condition	Cualquiera	Activo	Sí	Requerido
10	Limpiar Status-Event	Cualquiera	Activo	No	Requerido
11	Escribir Status-Event	Reposo	Activo	No	Requerido
12	Leer Status-Event	Cualquiera	Activo	Sí	Requerido
13-127	Reservado	-	-	-	-
128-255	Abierto Fabricantes	-	-	-	-

Tabla 5: Funciones dentro de la clase de los CommonCommands.

Todas estas funciones de comando (que se encuentran dentro de la clase CommonCommands) realizan operaciones desde la petición y modificación de TEDS, actualizar la caché de las TEDS que se encuentra en el NCAP, controla y gestiona los estados, las máscaras para iniciar tares…

Se puede observar que encada una de las dos tablas anteriores, la primera columna indica los parámetros *cmdClassId* y *cmdFunctionId* respectivamente, estos son los identificadores que se envían en la estructura de mensaje explicada anteriormente.

Los comandos más importantes son CommonCommands, Transducer operating state (XdcrOperate) y TIM active state commands (TIMActive).

Para llevar a cabo el comando deseado, se hace uso de las APIs de IEEE 1451.

*6.4.*APIs de IEEE 1451.0

Una interfaz de programación de aplicaciones (Application Program Interface, **API**) es el conjunto de funciones y métodos que ofrece una serie de recursos, a modo de biblioteca, para ser utilizado por otro software (otra capa distinta del protocolo, por ejemplo) como una capa de abstracción. Una API representa una interfaz de comunicación entre componentes de software.

Se trata del conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a ciertos servicios desde los procesos y representa un método para conseguir abstracción en la programación, generalmente entre los niveles o capas inferiores y los superiores del software.

En este estándar se definen dos tipos de APIs.

El TSI (Transducer Service Interface) es un API que se encuentra sólo en el NCAP y es usado para acceder a la capa correspondiente de IEEE 1451.0. Contiene métodos de lectura y escritura sobre TransducerChannels, TEDS y enviar comandos de configuración y control a los TIMs.

El Módulo Interfaz de Comunicaciones (Module Communications Interface) es un interfaz entre este estándar (1451.0) y otro estándar miembro de la familia

(como 1451.5, 1451.2, 1451.3, 1451.6). Es un interfaz simétrico ya que se encuentra implementado en ambas caras del NCAP y del TIM (entre las cuales se produce la comunicación de acuerdo a un estándar de la familia nombrado anteriormente).

Interface Definition Language, **IDL**, es un lenguaje informático utilizado para describir la interfaz de componentes software. Describe una interfaz en un lenguaje neutral.

El lenguaje IDL de una operación descrito en el estándar sólo usa tipos de datos definidos dentro de este estándar. Todos los datos específicos IDL dentro del estándar son prologados con "**IDL:**" en negrita, para facilitar la extracción automatizada de una copia del estándar.

IEEE1451Dot0

IDL: module IEEE1451Dot0 {};

Todas la interfaces IEEE 1451.0 están dentro del módulo IDL "IEEE 1451 Dot0". Se podría decir que la "Dot0" son las partes internas que conforman y definen la capa 1451.0 donde se encuentran tanto las interfaces como las funciones en las que se apoyan éstas.

La siguiente tabla ilustra la estructura top-level de IEEE 1451.0 API. Se define "IEEE1451Dot0Se puede observar su jerarquía en la siguiente figura.

Módulo	Descripción
TransducerServices	Esta es la API pública que las aplicaciones de medición y control usan para interactuar con la capa IEEE 1451.0 del estándar. Contiene clases e interfaces para descubrir TIMs, registrarlos, acceso a TransducerChannels para realizar mediciones o escribir actuadores, gestionar el acceso de TIMs, y de la lectura y la escritura de TEDS.
ModuleCommunications	Esta es la API que utiliza IEEE 1451.X para que se produzcan las comunicaciones entre el TIM y el NCAP. Tanto las interfaces "punto a punto"(P2P) y "red"(Net)como las respuestas se especifican
Args	Este paquete contiene todos los argumentos de IEEE 1451.0. La estructura de datos ArgumentArray se define.
Utils	Este paquete contiene las utilidades de clases e interfaces de utilidad para la conversión de un ArgumentArrays un OctetArrays (y viceversa). Esto se realiza a través de la interfaz Codec. Para implementaciones de IEEE 1451.X que necesiten cambiar la forma de codificación necesitarán registrar alternativos Codecs.

Tabla 6: Las APIs y sus recursos del IEEE1451.0

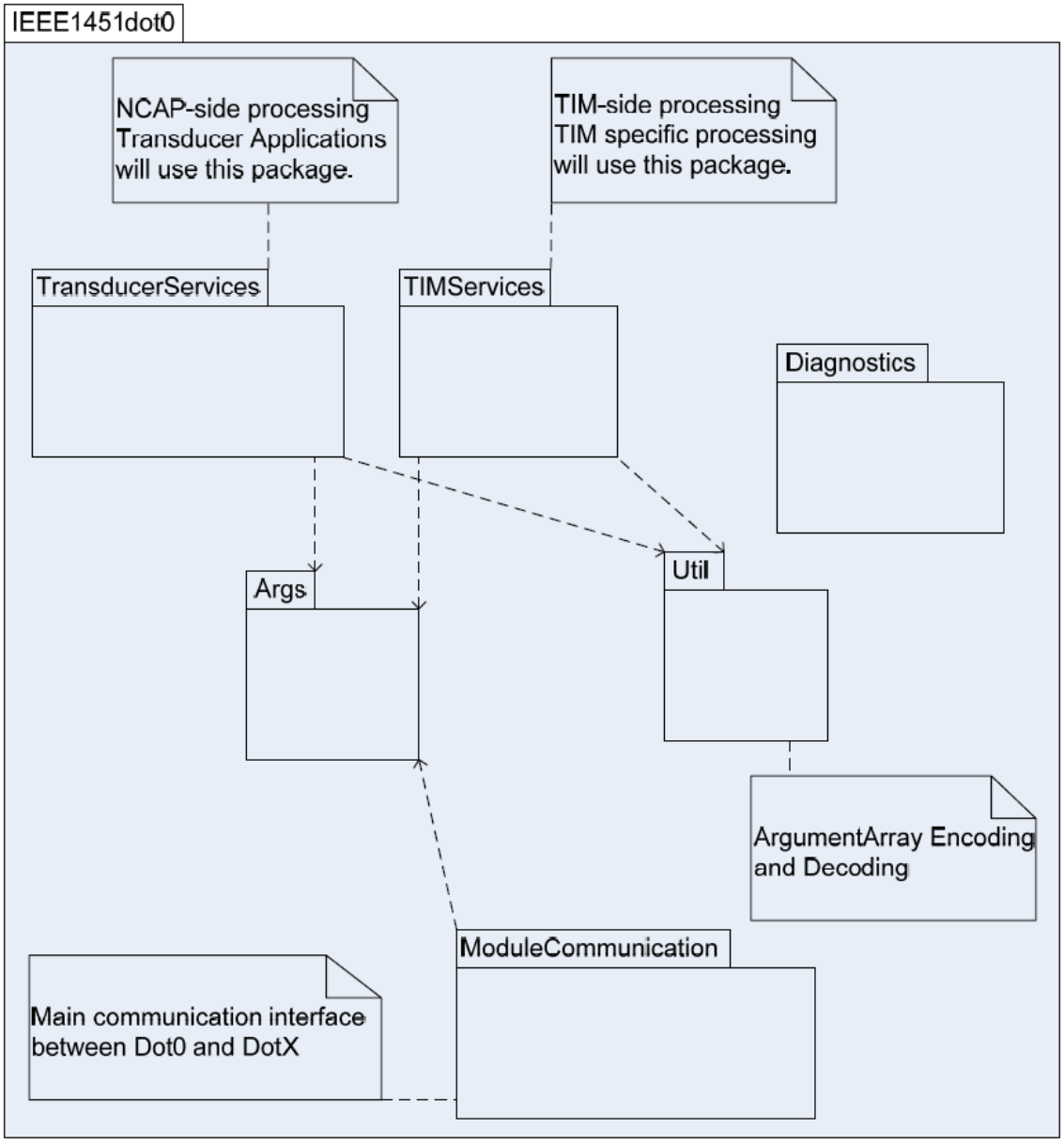


Ilustración 5: IEEE1451dot0.Relación entre las APIs

IEEE1451Dot0::Args

El módulo ARGS contiene los tipos de datos fundamentales y especiales en los que se van apoyar tanto la API de TransducerServices como la ModuleCommunication, son parte de los recursos a los que van a acudir dichas APIs.

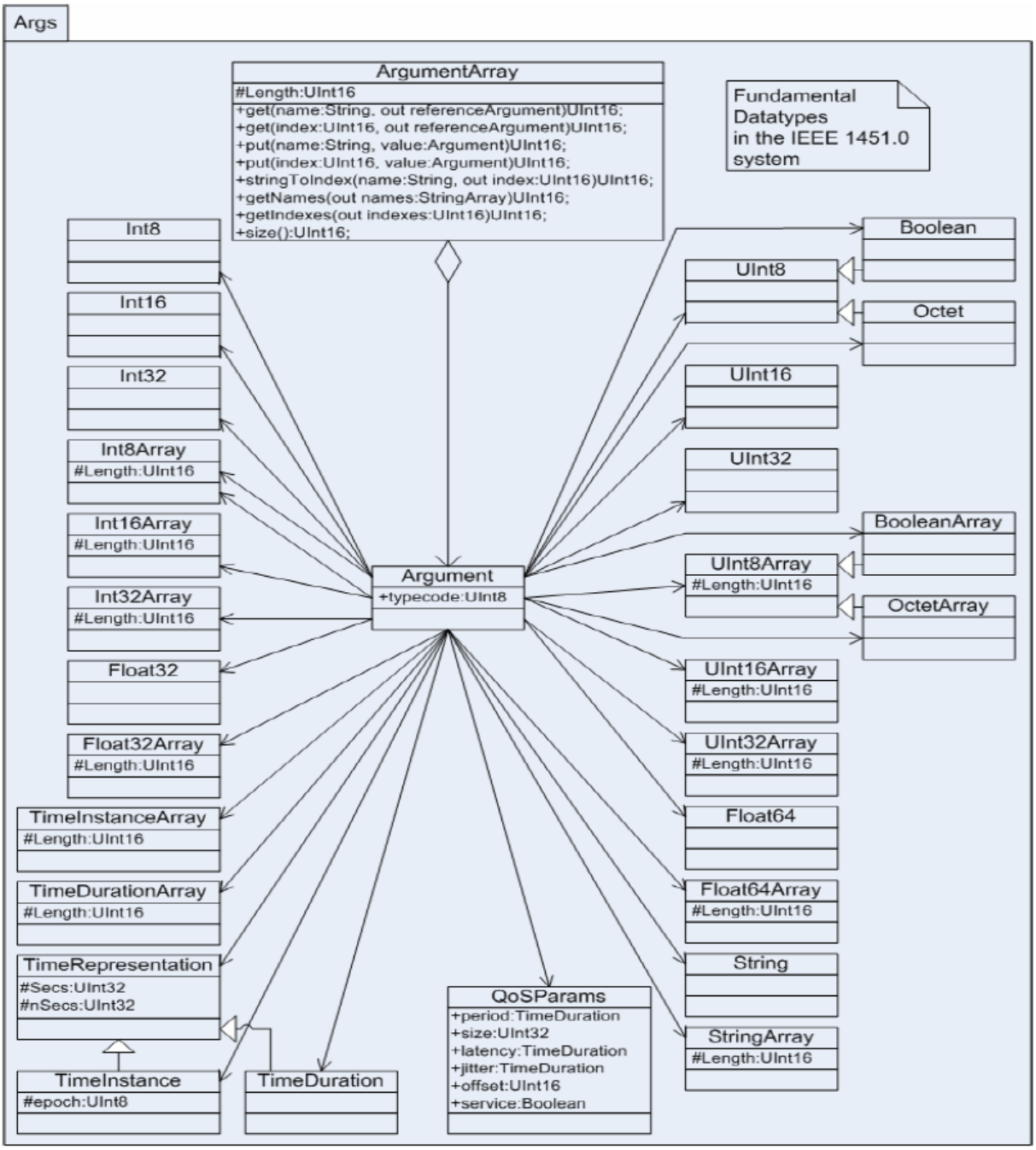


Ilustración 6: Argumentos y tipos de datos de IEEE 1451.0

Como se observa en la figura anterior, se especifican los tipos de datos y sus respectivos nombres, es muy importante respetar ambas cosas en una implementación.

6.5.Transducer Services API

IDL: module TransducerServices {};

La TransducerServices API proporciona la interfaz entre las aplicaciones que se ejecutan en el NCAP y las funciones definidas por el estándar IEEE 1451.0.

El módulo Transducer Services es subdividido a su vez en seis interfaces mostrados en la siguiente tabla.

Interfaz	Descripción
TimDiscovery	Este interfaz contiene métodos para encontrar módulos de comunicaciones IEEE 1451.X, TIMs y TransducerChannels.
TransducerAccess	Este interfaz proporciona métodos cuando una aplicación desea tener acceso a sensores o actuadores TransducerChannels.
TransducerManager	Las aplicaciones que necesitan más control del acceso sobre TIMs usarán métodos de este interfaz.
TedsManager	Para usos o aplicaciones en las que se necesita leer y/o escribir TEDS, gestión de TEDS.
CommManager	Se necesita acceso al módulo de comunicaciones del dispositivo.
AppCallback	Cuando ciertos usos o aplicaciones necesiten de opciones avanzadas. IMPORTANTE: en este estándar no se define ningún AppCallback. Está pensado para poder personalizar la red y para futuras modificaciones si hiciera falta.

Tabla 7: Interfaces que componen el Transducer Services API.

IEEE1451Dot0::TransducerServices::TimDiscovery

La interfaz TimDiscovery es ofrecida por la capa IEEE 1451.0 y llamada por la aplicación para proporcionar una forma común para "descubrir" y tener disponibles TIMs y TransducerChannels.

Los métodos que contiene TimDiscovery están en la siguiente tabla.

IEEE1451dot0::TransducerServices::TimDiscovery
Args::UInt16 reportCommModule(out Args::UInt8Array moduleIds);
Args::UInt16 reportTims(in Args::UInt8 moduleId, out Args::UInt16Array timIds);
Args::UInt16 reportChannels(in Args::UInt16 timId, out Args::UInt16Array channelIds, out Args::StringArray names);

Tabla 8: Funciones y métodos de la Interfaz TIM Discovery.

IEEE1451Dot0::TransducerServices::TimDiscovery::reportCommModule

Este método presenta los módulos de comunicaciones disponibles que están registrados con este estándar, es decir, muestra los Módulos de Comunicaciones registrados dentro de un NCAP. En el método *registerModule ( )*, la capa IEEE 1451.X llamará a la capa IEEE 1451.0 cuando esté lista para la operación y la capa IEEE 1451.0 asignará un "*moduleId*" único para cada Módulo de Comunicación de IEEE 1451.X. Tenga en cuenta que el NCAP puede tener:

- Una única interfaz IEEE 1451.X de una tecnología dada
- Múltiples interfaces de la misma tecnología (por ejemplo, IEEE 1451.2-RS232 en COM1 yCOM2).
- IEEE 1451.X múltiples interfaces de diferentes tecnologías

IEEE1451Dot0::TransducerServices::TimDiscovery::reportTims

Esta función devuelve un informe de los TIMs conocidos para un Módulo de Comunicaciones (*moduleId*) dado.

IEEE1451Dot0::TransducerServices::TimDiscovery::reportChannels

Esta función devuelve una lista sobre los TransducerChannels con sus respectivos nombres, para un TIM especificado. Dicha información se obtiene de la caché de las TEDS.

IEEE1451Dot0::TransducerServices::TransducerAccess

La interfaz de TransducerAccess es proporcionada por la capa IEEE 1451.0 y es llamada por la aplicación para proporcionar acceso a los"TransducerChannels". Para la mayoría de las solicitudes, que serán principalmente la interacción con esta interfaz para realizar TIM lectura y escritura. Para mantener este interfaz pequeño, los métodos más avanzados se encuentran en la interfaz de TransducerManager.

IEEE1451dot0::TransducerServices::TransducerAccess
Args::UInt16 open(in Args::UInt16 timId, in Args::UInt16 channelId, out Args::UInt16 transCommId);
::UInt16 openQoS(in Args::UInt16 timId, in Args::UInt16 channelId, inout Args::QoSParams qosParams, out Args::UInt16 transCommId);
Args::UInt16 openGroup(in Args::UInt16Array timIds, in Args::UInt16Array channelIds, out Args::UInt16 transCommId);
Args::UInt16 openGroupQoS(in Args::UInt16Array timIds, in Args::UInt16Array channelIds, inout Args::QoSParams qosParams, out Args::UInt16 transCommId);
Args::UInt16 close(in Args::UInt16 transCommId);
Args::UInt16 readData (in Args::UInt16 transCommId, in Args::TimeDuration time-out, in Args::UInt8 SamplingMode, out Args::ArgumentArray result);
Args::UInt16 writeData (in Args::UInt16 transCommId, in Args::TimeDuration time-out, in Args::UInt8 SamplingMode, in Args::ArgumentArray value);
Args::UInt16 startReadData(in Args::UInt16 transCommId, in Args::TimeInstance triggerTime, in Args::TimeDuration time-out, in Args::UInt8 SamplingMode, in AppCallback callback, out Args::UInt16operationId);
Args::UInt16 startWriteData(in Args::UInt16 transCommId, in Args::TimeInstance triggerTime, in Args::TimeDuration time-out, in Args::UInt8 SamplingMode, in Args::ArgumentArray value, in AppCallback callback, out Args::UInt16 operationId);
Args::UInt16 startStream(in Args::UInt16 transCommId, in AppCallback callback, out Args::UInt16 operationId);
Args::UInt16 cancel(in Args::UInt16 operationId);

Tabla 9: Funciones y métodos de la Interfaz Transducer Access.

IEEE1451Dot0::TransducerServices::TransducerAccess::open

Este método abre un canal de comunicación con el TIM/TransducerChannel deseado y devuelve un "transCommId" que será utilizado en las llamadas posteriores.

IEEE1451Dot0::TransducerServices::TransducerAccess::openQoS

Este método abre un canal de comunicación al deseado TIM/TransducerChannel y devuelve un "transCommId" que será utilizado en las llamadas posteriores. Si la llamada falla, el "qosParams" será modificado y devuelto a la aplicación, a fin de proporcionar una "pista" sobre que QoS pueden ser aceptables.

IEEE1451Dot0::TransducerServices::TransducerAccess::openGroup

El objetivo de este método consiste en asignar un número de TIM para direcciones de multidifusión que luego serán utilizados para asignar TransducerChannels a AddressGroups.

IEEE1451Dot0::TransducerServices::TransducerAccess::openGroupQoS

El objetivo de este método consiste en asignar un número de TIM para direcciones de multidifusión que luego serán utilizados para asignar TransducerChannels a AddressGroups.

IEEE1451Dot0::TransducerServices::TransducerAccess::close

Este método cierra una sesión de comunicación. La solicitud deberá considerar el transCommId como inválido. Hay que tener en cuenta que una posterior llamada a la función *open* puede devolver el valor anterior.

IEEE1451Dot0::TransducerServices::TransducerAccess::readData

Este método realiza una lectura bloqueante de un TransducerChannel especificado. El ArgumentArray puede tener muchos atributos, donde cada atributo está representado por un argumento distinto en el ArgumentArray. La aplicación puede controlar los atributos que se devuelven a través del uso de la llamada TransducerManager::configureAttributes( ).

En los casos en que sea una lectura de un único TransducerChannel, siempre habrá en el resultado un parámetro que contiene la lectura del TransducerChannel. El tipo de este argumento es determinado por el modelo de la TransducerChannel y si el TransducerChannel tiene asociada una Calibration TEDS. Por ejemplo, la lectura de un TransducerChannel simple, que no precisa una Calibration TEDS, siempre estará en la TransducerChannel (formato básico) (por ejemplo, Uint8 o Float32Array). Si el TransducerChannel especifica corrección en el NCAP a través de la Calibration TEDS, el tipo de dato siempre será float32 o Float32Array.

En los casos en que se trate de una lectura de un grupo de TransducerChannels, siempre será un "resultado" anidado ArgumentArray que contenga un argumento para cada TransducerChannel del grupo. Se accederá en orden, empezando con la posición del array "0". Esta organización corresponde a la orden de duplas de TIM/TransducerChannel en el OpenGroup () o openGroupQoS () de llamada.

IEEE1451Dot0::TransducerServices::TransducerAccess::writeData

Este método realiza una escritura bloqueante en un TransducerChannel especificado. El ArgumentArray tiene diversos atributos (Commands Messages), y cada atributo será representado por un argumento distinto en el ArgumentArray.

En caso de que se realice la escritura sobre un único TransducerChannel, el emisor deberá presentar un "valor", argumento que contiene el valor de la TransducerChannel. El emisor debe proporcionar el resultado en un tipo de dato compatible con el formato que requiere el TransducerChannel como se especifica en la TransducerChannelTEDS. La capa de IEEE 1451.0 en el NCAP llevará a cabo conversiones sencillas entre todos los tipos de datos numéricos.

IEEE1451Dot0::TransducerServices::TransducerAccess::startReadData

Este método comienza una lectura no-bloqueante de un TransducerChannels especificado. Cuando la lectura se completa, el AppCallback::measurementUpdate() será llamado en el objeto de la devolución de la llamada.

IEEE1451Dot0::TransducerServices::TransducerAccess::startWriteData

Este método comienza una escritura no-bloqueante de un TransducerChannels especificado. Cuando la escritura se completa, el AppCallback::actuationComplete() será llamado en el objeto de devolución de la llamada.

IEEE1451Dot0::TransducerServices::TransducerAccess::startStream

Este método comienza a tomar medidas de un Stream. El transCommId es el creado en cualquier llamada de los openQoS() o de openGroupQoS(). En este último caso, todos los TransducerChannels serán del mismo TIM. Cada vez que se disponga de nuevas medidas del Stream, la AppCallback::measurementUpdate() será llamada en el objeto de devolución de la llamada.

IEEE1451Dot0::TransducerServices::TransducerAccess::cancel

Este método cancelará una lectura, escritura o Stream bloqueante. La devolución de la llamada se hace con un error de código con estado CANCEL.

IEEE1451Dot0::TransducerServices::TransducerManager

La interfaz TransducerManager es proporcionada por IEEE 1451.0 y es llamada por la aplicación para facilitar el acceso a funciones más avanzadas. La mayoría de las aplicaciones no van a interactuar con este interfaz, será principalmente la interacción con la interfaz de TransducerAccess para realizar operaciones de lectura y escritura en el TransducerChannel. En la interfaz TransducerManager se encuentran los métodos avanzados con el fin de mantener al TransducerAccess con un pequeño tamaño.

IEEE1451dot0::TransducerServices::TransducerManager
Args::UInt16 lock(in Args::UInt16 transCommId, in Args::TimeDuration time-out);
Args::UInt16 unlock(in Args::UInt16 transCommId);
Args::UInt16 reportLocks(out Args::UInt16Array transCommIds);
Args::UInt16 breakLock(in Args::UInt16 transCommId);
Args::UInt16 sendCommand(in Args::UInt16 transCommId, in Args::TimeDuration time-out, in Args::UInt8 cmdClassId, in Args::UInt8 cmdFunctionId, in Args::ArgumentArray inArgs, out Args::ArgumentArray outArgs);
Args::UInt16 startCommand(in Args::UInt16 transCommId, in Args::TimeInstance triggerTime, in Args::TimeDuration time-out, in Args::UInt8 cmdClassId, in Args::UInt8 cmdFunctionId, in Args::ArgumentArray inArgs, in AppCallback callback, out Args::UInt16 operationId);
Args::UInt16 configureAttributes(in Args::UInt16 transCommId, in Args::StringArray attributeNames);
Args::UInt16 trigger(in Args::UInt16 transCommId, in Args::TimeInstance triggerTime, in Args::TimeDuration time-out, in Args::UInt16 SamplingMode);
Args::UInt16 startTrigger(in Args::UInt16 transCommId, in Args::TimeInstance triggerTime, in Args::TimeDuration time-out, in Args::UInt16 SamplingMode, in AppCallback callback, out Args::UInt16 operationId);
Args::UInt16 clear(in Args::UInt16 transCommId, in Args::TimeDuration time-out, in Args::UInt8 clearMode);
Args::UInt16 registerStatusChange(in Args::UInt16 transCommId, in Args::TimeDuration time-out, in AppCallback callback, out Args::UInt16 operationId);
Args::UInt16 unregisterStatusChange(in Args::UInt16 transCommId);

Tabla 10: Funciones y métodos de la Interfaz Transducer Manager.

IEEE1451Dot0::TransducerServices::TransducerManager::lock

Este método bloqueará un TIM o TransducerChannels específico representado por el transCommId. De esta manera se evitará que otras aplicaciones tengan acceso a esos recursos. Es responsabilidad de la aplicación bloquear los recursos para evitar puntos muertos en entornos multi-threaded.

IEEE1451Dot0::TransducerServices::TransducerManager::unlock

Este método desbloquea el TIM/TransducerChannels representado por el transCommId. Esto permitirá que otras aplicaciones puedan acceder a esos recursos.

IEEE1451Dot0::TransducerServices::TransducerManager::reportLocks

Este método da un listado de todos los transCommIds que actualmente se encuentren cerrados/bloqueados.

IEEE1451Dot0::TransducerServices::TransducerManager::breakLock

Este método termina con un cierre/bloqueo. Si una operación no-bloqueante como una lectura, escritura o Stream de medición, estaba en curso, la devolución de llamada manda el código de error apropiado.

IEEE1451Dot0::TransducerServices::TransducerManager::sendCommand

Este método realiza una operación bloqueante. El formato de entrada y salida son argumentos "*commanddependent*". Cuando se llame a dicho método hay que asegurarse de utilizar los tipos de datos correctos para cada argumento de la entrada.

Si se trata de un comando personalizado, la aplicación debe utilizar "Command TEDS" conteniendo el octetArray que tiene el comando.

IEEE1451Dot0::TransducerServices::TransducerManager::sendCommandRaw

Este método es como el anterior, pero el envío de datos se hace en "crudo". Información que posteriormente hay que tratar o cuya funcionalidad está fuera del estándar y por tanto hay que tratarla.

IEEE1451Dot0::TransducerServices::TransducerManager::startCommand

Este método inicia una operación no-bloqueante. El formato de los argumentos de entrada es "*command dependents*". Cuando se produzca la llamada hay que asegurarse de utilizar los tipos de datos correctos para cada argumento de entrada.

IEEE1451Dot0::TransducerServices::TransducerManager::configureAttributes

Este método configura un "*transCommId*" para leer o escuchar las operaciones de medición. En él se especifica que atributos deben ser devueltos en el ArgumentArray. En el apartado de "*Comandos*" y "*Estructuras de los Mensajes*" se da más información sobre los nombres de dichos atributos.

IEEE1451Dot0::TransducerServices::TransducerManager::trigger

Este método realiza un trigger bloqueante sobre un *transCommId* especificado.

IEEE1451Dot0::TransducerServices::TransducerManager::startTrigger

Este método realiza un trigger no-bloqueante sobre un *transCommId* especificado.

IEEE1451Dot0::TransducerServices::TransducerManager::clear

Este método borra un *transCommId* especificado.

IEEE1451Dot0::TransducerServices::TransducerManager::registerStatusChange

Este método registra una aplicación *callBack* para modificar los eventos de cambio de estado en el *transCommId* de un TIM especificado.

IEEE1451Dot0::TransducerServices::TransducerManager::unregsterStatusChange

Este método anula el registro de una aplicación *callBack* para modificar los eventos de cambio de estado en el *transCommId* de un TIM especificado.

IEEE1451Dot0::TransducerServices::TedsManager

TedsManager es una interfaz proporcionada por la capa IEEE 1451.0 y se llama por la aplicación para tener acceso a las TEDS.

IEEE1451Dot0::TransducerServices::TedsManager
Args::UInt16 readTeds(in Args::UInt16 transCommId, in Args::TimeDuration time-out, in Args::UInt8 tedsType,out Args::ArgumentArray teds);
Args::UInt16 writeTeds(in Args::UInt16 transCommId, in Args::TimeDuration time-out, in Args::UInt8 tedsType,in Args::ArgumentArray teds);
Args::UInt16 readRawTeds(in Args::UInt16 transCommId, in Args::TimeDuration time-out, in Args::UInt8tedsType, out Args::OctetArray rawTeds);
Args::UInt16 writeRawTeds(in Args::UInt16 transCommId, in Args::TimeDuration time-out, in Args::UInt8tedsType, in Args::OctetArray rawTeds);
Args::UInt16 updateTedsCache(in Args::UInt16 transCommId, in Args::TimeDuration time-out, in Args::UInt8tedsType);

Tabla 11: Funciones y métodos de la Interfaz Teds Manager.

IEEE1451Dot0::TransducerServices::TedsManager::readTeds

Este método realiza la lectura de una TEDS especificada de la caché TEDS. Si la TEDS no está disponible en la caché, se leerá del TIM. La información TEDS se devuelve en un *ArgumentArray*.

IEEE1451Dot0::TransducerServices::TedsManager::writeTeds

Este método realiza la escritura de una TEDS en el TIM. La caché TEDS también se actualiza y la almacena si la escritura se realiza correctamente. La información proporcionada, está codificada en un *ArgumentArray*. Será convertida internamente a la forma correcta "tupla" (terna) y será transferido al TIM en un OctetArray.

IEEE1451Dot0::TransducerServices::TedsManager::readRawTeds

Este método realiza la lectura de una TEDS sin pasar por la caché de TEDS, es decir, sin tener en cuenta en ningún momento a la caché de las TEDS. La información se devuelve en forma de OctetArray, sin tratar la información y la caché TEDS no se actualizará.

IEEE1451Dot0::TransducerServices::TedsManager:: writeRawTeds

Este método realiza la escritura de una TEDS en el TIM sin pasar por la caché de las TEDS. La información proporcionada está codificada en forma de "tupla" en un OctetArray. Hay que tener mucho cuidado al rellenar todos los campos y al crear la tupla (TLV).

IEEE1451Dot0::TransducerServices::TedsManager::updateTedsCache

Este método actualiza la caché TEDS. La checksum de la TEDS en cuestión del TIM se compara con la misma checksum de la caché TEDS, si las checksums son diferentes, las TEDS se leerán desde el TIM y se almacenarán en la caché TEDS.

IEEE1451Dot0::TransducerServices::CommManager

La interfaz CommManager es proporcionada por la capa IEEE 1451.0 y es llamada por la aplicación para crear un mecanismo común para gestionar las comunicaciones disponibles en una NCAP.

IEEE1451dot0::TransducerServices::CommManager
Args::UInt16 getCommModule(in Args::UInt8 moduleId, out ModuleCommunication::Comm commObject, out Args::UInt8 type, out Args::UInt8 technologyId);

Tabla 12: Funciones y métodos de la Interfaz CommManager.

IEEE1451Dot0::TransducerServices::CommManager::getCommModule

Este método devuelve el resumen del objeto "Comm" (cómo es la comunicación) para aplicaciones que necesitan pasar por alto el procesamiento que hace IEEE 1451.0 e interactuar directamente con las comunicaciones subyacentes al objeto. Al consultar el "tipo" de parámetros, la aplicación puede llegar de forma segura hasta cualquiera de los dos objetos, ya sea el "P2PComm" o el "Netcomm".

IEEE1451Dot0::TransducerServices::AppCallback

La interfaz AppCallback es proporcionado por las aplicaciones y es llamado por el IEEE 1451.0 capa para proporcionar l acceso a los no-bloqueante de E / S y la medición arroyos

6.6.MÓDULO DE COMUNICACIONES API

6.6.1. Introducción

El módulo de comunicaciones, como se ha dicho anteriormente, es un elemento básico para desarrollar el estándar IEEE 1451. Este módulo es el responsable de hacer posible la comunicación entre un NCAP y un TIM (en ocasiones TIM-TIM) y viceversa. Para que ello ocurra, está claro que tanto el TIM como el NCAP deben de tener implementado dicho módulo y además debe de ser simétrico, es decir, las funciones y operaciones que realiza uno de ellos el otro el otro módulo de comunicaciones la “deshace”.

Dentro de dicho módulo se encuentran las interfaces que se van a encargar de “traducir” los comandos de “transducer services de 1451.0” y mediante las cuales

se va acceder a la capa de convergencia, que es la responsable de entregar la información, primero a 1451.X (la parte del estándar que se vaya a encargar del transporte como 1451.2, 1451.3, 1451.5…) y una vez dado este paso a la tecnología responsable de realizar el envío de la información desde el NCAP al TIM (o viceversa)

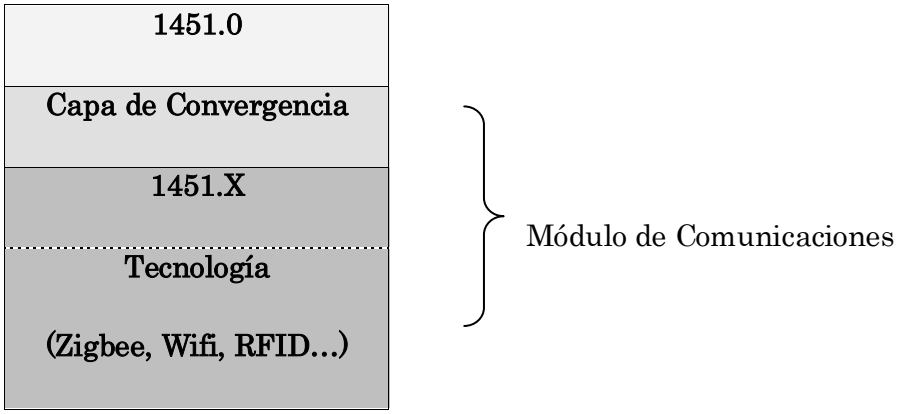


Ilustración 7: Capas desde el nivel físico hasta los servicios del 1451.0.

Es importante pararse detenidamente en este punto.

En la especificación del estándar IEEE 1451.0 se introduce el concepto del *Módulo de Comunicaciones* así como su utilidad. Es muy importante señalar también que además de esto, dicha especificación recoge los tipos de interfaces con los nombres que deben tener, con los parámetros de entrada/salida y con las funciones y métodos que se encuentran dentro de dichas interfaces. De esta forma estamos asegurando que la estructura de las interfaces sea común independientemente de cómo y con qué tecnología realicemos la comunicación.

Una vez dicho esto, cada especificación IEEE 1451.X (1451.2, 1451.3, 1451.5, 1451.6, P1451.7…) será la responsable de otorgar la capa de transporte respetando la estructura de las interfaces que impone IEEE 1451.0. La *capa de convergencia* (la cual se especifica dentro de su respectivo 1451.X) no es sólo el punto en el que convergen IEEE 1451.0 y 1451.X, sino que además sienta las bases para entregar de la forma adecuada la información a la tecnología responsable de realizar la comunicación (Bluetooth, Zigbee..).

En el caso que concierne a este documento, IEEE 1451.5, y más concretamente *Zigbee*, se verá parte de cómo se realiza estos pasos más delante de este documento.

6.6.2. Aspectos y condiciones generales

Como se ha dicho anteriormente el Módulo de comunicaciones posibilita la comunicación NCAP-TIM, TIM-NCAP y TIM-TIM (opcional y en los casos que sea soportado). Ahora bien, existen cuatro aspectos a tener en cuenta de cómo puede ser dicha comunicación:

- **One-Way** o **Two-Way**
    - *One-Way* se produce cuando el transmisor envía un mensaje al receptor terminando la comunicación. *Two-Way*, por su parte, hace que el transmisor envíe un mensaje al receptor y éste a su vez le contesta.
- **One-to-one** o **One-to-many**
    - *One-to-one* se produce cuando dos dispositivos, transmisor y receptor, participan en la comunicación. Por su parte, *One-to many* es cuando el transmisor se comunica con un grupo de receptores.
- **Default-QoS** o **Special-QoS** (QoS, Quality of Service)
    - Cuando no se indique de otra manera, la calidad en la comunicación será *Default-QoS*, es decir, best-effort. Si por lo contrario se desea que cumpla algún parámetro será *Special-QoS*

- **Point-to-Point** o **Network**
    - La comunicación será *Point-to-Point* cuando el interfaz físico sea simple y para un dispositivo, y por su parte, se denominará *Network* cuando el medio sea compartido por varios dispositivos.

El Módulo de Comunicaciones es subdividido en 3 interfaces:

- **Comm**: Este interfaz proporciona mecanismo y métodos para controlar el ciclo de vida de 1451.X.
- **Registration**: Este interfaz contiene métodos que te permiten registrar módulos IEEE 1451.X con la capa IEEE 1451.0.
- **Receive**: No hay definidos métodos generales para este interfaz. Se establece para futuras expansiones.

Cada uno de estos interfaces y como se verá más detenidamente un poco más adelante, se divide a su vez en dos interfaces: P2P (para comunicaciones directas entre un NCAP y TIM) y Net (Comunicaciones entre NCAP y múltiples TIMs), según sea uno u otro en función del cuarto aspecto mencionado anteriormente.

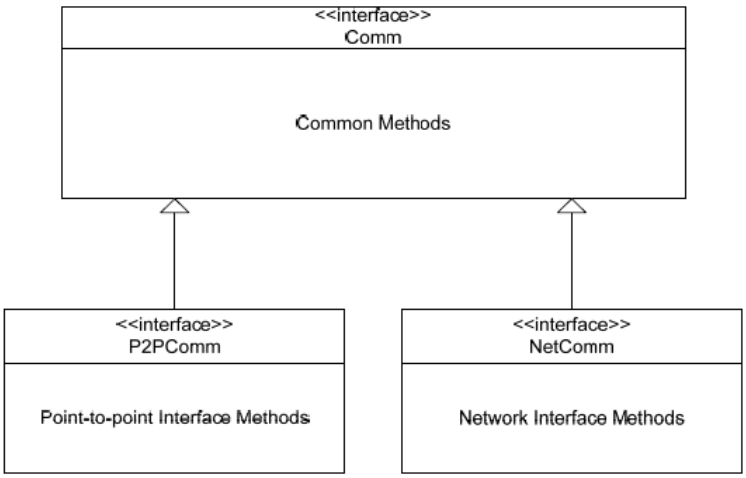


Ilustración 8: Interfaz Comm.

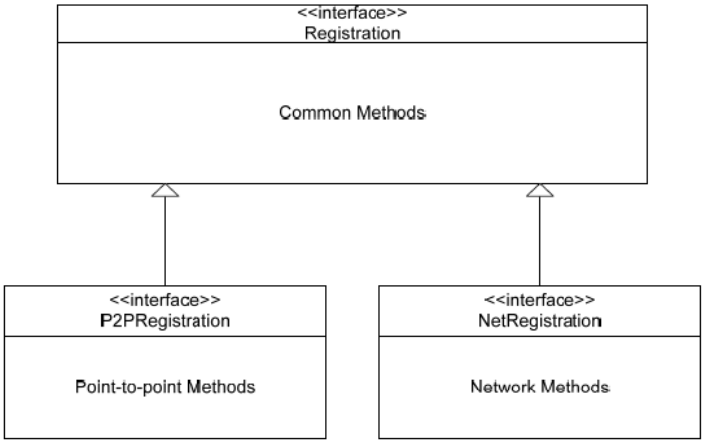


Ilustración 9: Interfaz registration.

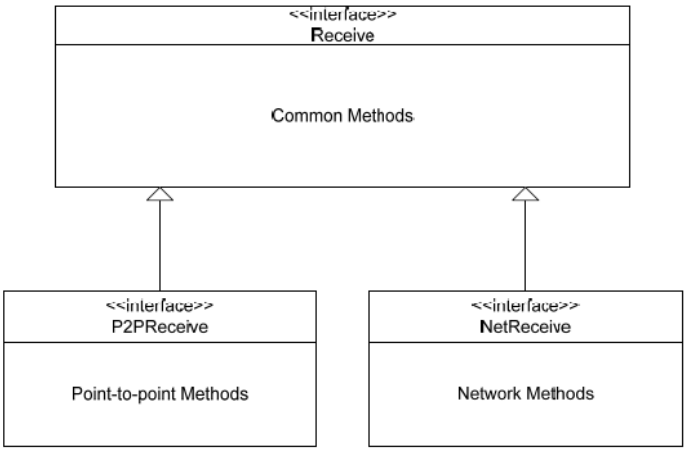


Ilustración 10: Interfaz Receive.

Como se ha mencionado anteriormente, estas APIs son simétricas. A continuación, se muestran los 3 patrones de comunicación contemplados:

- Two-way/One-to-one: por ejemplo, un NCAP solicita el comando para leer una TEDS a un TIM específico y dicho TIM le contesta con el contenido correspondiente.
- One-way/One-to-one: por ejemplo, un TIM genera una medida periódica y la envía a un NCAP.

- One-way/One-to-many: por ejemplo, igual que caso anterior pero participando varios TIMs en la medida.

A la hora de implementar la red de sensores hay que hacer una serie de consideraciones previas. En el caso del *Módulo de Comunicaciones*, se debe conocer el grado versatilidad y complejidad que te hace falta. Si se busca un TIM sencillo, el cual sólo tenga que contestar peticiones de NCAP o mandar comandos periódicamente puede que no sea necesario que implemente las interfaces *"Net"* y que sea suficiente con *"P2P"*, de esta forma se ahorra bastante complejidad en las interfaces pero tener en cuenta que un TIM no puede iniciar una comunicación. Si por el contrario lo que se busca es una red que te permita comunicarte a través de un único módulo de comunicaciones de un NCAP con muchos TIMs, hacen falta los interfaces *"Net"*.

Importante tener en cuenta que los NCAP o incluso los TIMs (aunque menos probable) pueden tener más de un *Módulo de Comunicaciones* si tienen implementados sensores con distinta tecnología.

Como resumen de lo anterior, si trabajo con un módulo de comunicaciones con interfaces P2P y además deseo que el NCAP se comunique con "**n**" TIMs hay que tener en cuenta que en el NCAP voy a tener "**n**" módulos de comunicaciones, uno para cada TIM. No obstante al utilizar un módulo de comunicaciones que trabaje con interfaz NET, voy a poder compartir el módulo de comunicaciones del NCAP para poder comunicarme con varios TIMs.

En las configuraciones NETCOMM, cada nodo de la red se identifica por unos parámetros de "nodo" de comunicación específicos de IEEE 1451.X. Por ejemplo, en 1451.5 802.11, cada nodo tiene una dirección IP única y puertos de red que se asignan para las comunicaciones IEEE 1451.0. Cada IEEE 1451.X puede tener diferentes parámetros en función de la tecnología que utilice.

Aunque la naturaleza de estos parámetros es específica de cada tecnología IEEE 1451.X, la capa IEEE 1451.0 necesita la capacidad de consultar cómo pasar

a 1451.X de una forma genérica. El getNodeParams() es utilizado para conseguir estos parámetros en un ArgumentArray. Esta llamada devuelve los parámetros de la capa de IEEE 1451.X locales.

Hay 3 parámetros muy importantes: "destId", commId y "msgId"

Interfaces.Module Communications API

IDL: module ModuleCommunication { };

Las interfaces del Módulo de Comunicaciones (Module Communications API) proporcionan y definen las funciones y métodos para convertir la información de los servicios IEEE 1451.0 a la capa física en la que se va a llevar a cabo la comunicación NCAP-TIM.

Interfaz	Descripción
Comm	Este interfaz proporciona mecanismo y métodos para controlar el ciclo de vida de 1451.X.
P2PComm	Contiene operaciones que te permiten establecer comunicaciones punto a punto.
NetComm	Contiene operaciones que te permiten establecer comunicaciones de red.
Registration	Este interfaz contiene métodos que te permiten registrar módulos IEEE 1451.X con la capa IEEE 1451.0.
P2PRegistration	Proporciona métodos para registrar TIMs específicos con la capa 1451.0.
NetRegistration	Proporciona métodos para registrar TIMs específicos y grupos de TIMs con la capa 1451.0.
Receive	No hay definidos métodos genéricos para este interfaz. Se establece para futuras expansiones.
P2PRecieve	Proporciona métodos el modulo de comunicaciones punto a punto de IEEE 1451.X que notifican a la capa 1451.0 que cierto mensaje haya llegado. También incluye métodos para abortar operaciones.
NetReceive	Proporciona métodos para el modulo de comunicaciones de red de IEEE 1451.X que notifican a la capa 1451.0 que cierto mensaje haya llegado. También incluye métodos para abortar operaciones.

Tabla 13: Interfaces que componen el modulo de comunicaciones.

En el caso de este proyecto, la primera cuestión para comenzar una implementación del módulo de comunicaciones era la elección de la forma en que se iba a producir dicha comunicación. En un primer momento, se valoró la posibilidad de implementar un sistema punto a punto (P2P) ya que tanto la cantidad de las funciones como la complejidad que éstas requerían era menor que mediante interfaz NET. No obstante, al desarrollar las primeras funciones se constató que no existía excesiva complejidad en las funciones del interfaz P2P respecto a las NET y lo que era más importante, el módulo de comunicaciones con el que trabajamos es

1451.5 (concretamente con tecnología Zigbee) respondía y tenía más sentido para trabajar en un sistema el cual nos permitiera hacer una red con dispositivos (NCAP con TIMs) y para ello había que recurrir a las funciones que contenía el interfaz de red (NET).

La interfaz "P2P" es más apropiada para situaciones en las que la comunicación se realice a por cable y con pocos TIMs (por ejemplo RS-232) especificadas en IEEE 1451.2.

Dentro de IEEE1451.5 la **capa de convergencia** (interfaz que contiene las funciones para pasar de 1451.0 a 1451.5 y viceversa) varía según la tecnología.

6.7.TEDS. TRANSDUCER ELECTRONIC DATA SHEETS

6.7.1. Introducción

Transducer Electronic Data Sheet (Hoja de Datos Electrónica del Transductor, TEDS) es un método estandarizado de almacenar datos como la identificación, la calibración, datos de corrección y la información relacionada con el fabricante de un transductor (sensor o actuador). Los formatos TEDS son definidos en la familia de estándares IEEE 1451 que describe un juego de interfaces de comunicación abiertos, comunes, independientes de red para conectar transductores a microprocesadores, sistemas de instrumentación y controlar redes. Sin lugar a dudas, las TEDS son la clave más potente del estándar IEEE 1451 que permiten a un sensor inteligente determinado "entenderse" dentro de una red con otros sensores.

Es muy importante entender que IEEE 1451 (salvo 1451.4 como se explica más adelante) no tiene TEDS (o plantillas para control de un sensor) específicas para cada tipo de sensor (termómetro, barómetro, presencia…) si no que dentro de una TEDS común para todo tipo de sensor se establece la magnitud física a medir,

rangos, sensibilidad… Este es uno de los motivos por los cuales es estándar IEEE 1451.

En IEEE 1451.4, las TEDS sí que cambian su filosofía respecto a las del resto del estándar. IEEE 1451.4 establece una "Basic TEDS" en el transductor la cual tiene una longitud de 64 bits y en la que se guarda información relativa al fabricante, modelo, versión, código de serie

Manufacturer ID	Model number	Version letter	Version number	Serial number
14 bits (17–16381) [a]	15 bits (0–32767)	5 bits Char (A–Z, data type Chr5)	6 bits (0–63)	24 bits (0–16777215)

Tabla 14: Contenido de las Basic TEDS.

En vez de trabajar con TEDS como el resto del estándar, establece una plantillas (templates) las cuales son específicas para cada tipo de sensor (termómetro, barómetro…) las cuales se alojan junto con las TEDS y éstas son la que hacen referencia a dichas plantillas. En el *Anexo A* de la especificación IEEE 1451.4 se encuentran todas las plantillas disponibles.

Una TEDS, en esencia, es un dispositivo de memoria conectado al transductor y contiene la información necesaria según un instrumento de medida o el sistema de control al interfaz con un transductor. De todas formas, una TEDS puede ser puesta en práctica de dos formas:

1. Primero, la TEDS puede residir en la memoria integrada, típicamente un EEPROM, dentro del mismo transductor que es conectado al instrumento de medida o el sistema de control.

2. Segundo, una TEDS virtual puede existir como un fichero de datos al que se pueda acceder según el instrumento de medida o el sistema de control. Es muy práctico donde la memoria integrada no puede estar disponible.

   Formato de TEDS para IEEE 1451.0

Todas las TEDS tienen el formato mostrado en la siguiente tabla.

El primer campo en cualquier TEDS es la longitud, formada por 4 octetos sin signo, dicha longitud indica el número total de octetos en el bloque de datos (*data block*) más los dos octetos del *checksum*.

El siguiente bloque es el contenido de la información de la propia TEDS. Dependiendo de dicha TEDS, la información puede estar en forma binaria o en formato texto.

El último campo es el checksum el cual vela por la integridad y detectar errores en una TEDS.

Field	Descripción	Tipo	Nº octetos
-	Longitud de TEDS	Uint32	4
1 a N	Bloque de Datos	Variable	Variable
-	Checksum	Uint16	2

Tabla 15: Campos que componen una TEDS.

Dentro del bloque de datos (*data block*) reside toda la información de los campos de la TEDS. Dicha información se almacena en ternas de datos (TLV, Type/Length/Value). En el bloque de datos hay tantas ternas (o tuplas) de estos datos como campos tenga la dicha TEDS.

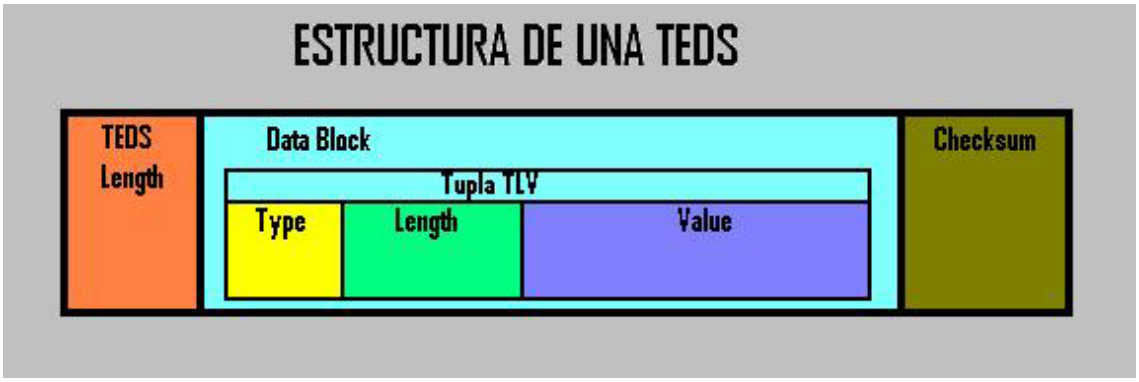


Ilustración 11: Estructura de una TEDS.

En el "*Data Block"* se alojan múltiples TLVs.

Campo	Descripción
Tipo	Este código identifica el campo en la TEDS que está contenida dentro del campo "Valor", es decir, indica qué dato es. Excepto cuando se almacene un 2 o 3, el mismo número en el campo de tipo tendrá un significado diferente en distintas TEDS.
Longitud	El número en este campo da el número de octetos en el campo "Valor". El número de octetos en el campo "Longitud" es controlado por una entrada en la TEDS Identificación TLV.
Valor	Contiene la información de la TEDS

Tabla 16: Descripción de los campos de una TLV.

En el caso en que la TEDS esté en formato texto, este Tipo/Longitud/Valor (TLV) es usado para proporcionar un directorio para dar acceso a las diferentes secciones del texto de la TEDS que usa XML para el contenido de la información.

En la utilización de estructuras TLVs, cada entrada se almacena como una terna TLV. El campo "Tipo" es una etiqueta de 1 octeto que identifica la TLV, similar a la función a HTML o etiquetas XML. El campo "Longitud" especifica el número de los octetos del campo "Valor", y el campo "Valor" son los datos, la información en sí. Cada entrada puede estar compuesta de uno o varios TLVs. La estructura, o el tipo de datos, del campo de valor son definidos en la especificación para TEDS en este estándar.

IMPORTANTE: COMPATIBILDADES TEDS

Es sumamente importante señalar que tanto IEEE 1451.2 como IEEE 1451.3 las TEDS son ligeramente distintas a las que se establecen en IEEE 1451.0 ya que las primeras TEDS se especificaron en 1997 a través de 1451.2 y que con la salida a la luz de 1451.0 diez años después supone una actualización de las mismas ya que en las primeras no se utilizan estructuras de ternas de datos TLV (no reconocidas por 1451.2 y1451.3).

En 1451.2 y 1451.3 el primer octeto a continuación del campo *longitud* de la Meta-TEDS siempre contiene el valor 2.A su vez, el primer octeto a continuación del campo *longitud* en cualquier TEDS IEEE 1451.0 es siempre el *tipo* de una TLV, y el valor 2 está reservado y no se utilizará. La Meta-TEDS es la única TEDS de IEEE 1451.2 y 1451.3 que contiene información sobre la versión de las TEDS empleada, por lo que es necesario que la Meta-TEDS sea la primera TEDS leída con el fin obtener información de la versión de la TEDS.

6.7.2. TEDS Identification Header

Este campo es obligatorio en todas las TEDS. Siempre que el valor *Type* de una TLV sea "3", el valor del campo "Value" hará referencia a la identificación de cabecera conteniendo información relativa de la TEDS a la familia del estándar que pertenece, versión y la clase. Además este campo siempre será el primero en una TEDS.

TLV		Descripción
Type		TEDS Identifier. Siempre vale 3 para todas las TEDS
Length		Indica que el campo Value va a tener una longitud de 4 octetos
Value	Familia	Identifica al miembro de IEEE 1451 que identifica la TEDS
	Clase	Este valor identifica la TEDS que está siendo requerida
	Versión	Identifica la versión de la TEDS, esto está más pensado para futuras expansiones o cambios en el estándar.
	Terna longitud	Este campo da la información de todos los octetos del campo Length de todas las TLVs de la TEDS (sin incluir esta TLV).

Tabla 17: Estructura de un identificador de cabecera de una TEDS.

6.7.3. Meta-TEDS

Meta-TEDS es una TEDS obligatoria. La función de una Meta-TEDS es hacer disponible al interfaz toda la información que necesita para tener acceso a cualquier TransducerChannel y la información común a todos los TransducerChannels. Esta TEDS tiene que estar alojada en el TIM al que identifique, es decir, la Meta-TEDS identifica e informa sobre un determinado TIM, no de un transductor. Debe de haber una de estas TEDS por cada TIM que haya en la red.

De forma general, contiene un identificador UUID, que sirve para identificar forma única a dicho TIM, también contiene parámetros de "worst timing

parameters" y *"time out values"* que van a servir para identificar cuando el TIM no está respondiendo.

A la hora de implementar esta Meta-TEDS hay que tener en cuenta de que se trata de información que no va cambiar, al menos a priori, en una implementación sencilla. Por ejemplo, y en la situación en la cual está pensado desarrollar este estándar (ez430-RF2480), tenemos un microprocesador de bajo consumo en el cual tenemos unas limitaciones de memoria, sobre todo de RAM, por lo que la gran parte de la información de esta TEDS puede ser alojada en la memoria FLASH por medio de constantes para así tener menor gasto de la memoria RAM.

Una Meta-TEDS ocupa, de forma general y como mínimo un total de **40 bytes**.

6.7.4. Transducerchannel TEDS

Se trata de una TEDS obligatoria. Aporta información detallada sobre un transductor específico, es la TEDS más importante o al menos a la que más veces se accede. Presenta la información desde el TIM hasta el transductor en cuestión, por lo que debe de existir una de estas TEDS por cada transductor. Da los parámetros físicos que se están midiendo o controlando, el rango sobre el cual el TransducerChannel opera, las características de la E/S digitales, el modo operativo de la unidad y la información de los tiempos.

También se especifica si el transductor es un sensor, un actuador o un sensor evento. TransducerChannel TEDS también contiene información sobre la calibración (aunque de forma sencilla) del transductor pero si es necesario una calibración más compleja y completa, esta TEDS indicará que la calibración de dicho transductor la realizará la Calibration TEDS.

El *"data block"* de esta TEDS ocupa, en torno a **136 bytes**.

6.7.5. User´s Transducer Name TEDS

Esta TEDS es obligatoria para los TIMs y es recomendable para todos los transductores. El User´s TransducerName TEDS proporciona un lugar para almacenar el nombre por el cual el sistema o el usuario final conocerán a este transductor. Este TEDS se establece para poder soportar "Object Tags" en IEEE 1451.1.

El *"data block"* de esta TEDS ocupa, como mínimo, **11 bytes**.

6.7.6. Phy TEDS

La PHY TEDS es una TEDS obligatoria. Contiene información sobre la medio físico por el cual se va a realizar la comunicación entre un TIM y un NCAP, dentro del Módulo de Comunicaciones. La función es hacer disponible al interfaz toda la información que necesita para tener acceso a cualquier canal, más la información común a todos los canales.

Importante señalar que esta TEDS la especifica la parte del estándar la cual se vaya a emplear como módulo de comunicaciones, es decir, el IEEE 1451.X correspondiente (como por ejemplo 1451.2, 1451.3, 1451.5 ,1451.6, p1451.7 y posteriores).

Al igual que ocurría con Meta-TEDS, a la hora de implementar esta TEDS se hará de forma que los datos sólo puedan ser leídos alojando dicha información en memoria ROM o Flash. El tamaño en bytes puede variar según el 1451.5 especificado, pero el *"data block"* ocupará, como término medio, **42 bytes** para IEEE 1451.5.

6.7.7. Calibration TEDS

Esta TEDS es opcional. Calibration TEDS contiene la información sobre los parámetros de calibración, y el intervalo de calibración de un transductor. También proporciona constantes necesarias para convertir los datos (obtenidos por un transductor) en unidades de la ingeniería para sensores o convertir unidades de la ingeniería a la forma requerida por un actuador. Las entradas en Calibration TEDS son usadas para la corrección.

Existen otras dos TEDS opcionales, FrequencyResponse TEDS y Transfer Function TEDS que se utilizan también para correcciones de datos y comunicación, las cuales se especifican a continuación.

6.7.8. Frequency response TEDS

Es una TEDS opcional. La función de FrequencyResponse TEDS es hacer disponible la información de acuerdo a la frecuencia del TransducerChannel para el usuario. FrequencyResponse TEDS proporciona una caracterización de la frecuencia y la respuesta en fase del TransducerChannel. Los factores que afectan a la respuesta frecuencial son el sensor, el acondicionamiento de la señal analógica, el filtro de anti-aliasing y el tratamiento de la señal digital.

La TEDS da la respuesta punto a punto desde sensor analógico hasta la salida digital. Esta caracterización asume que la salida del TIM se lee a una velocidad soportada por la respuesta frecuencial.

La FrequencyResponse TEDS es normalizada a la frecuencia de referencia, es decir, se asume que los datos de transductor son referidos a esta frecuencia.

6.7.9. Transfer Function TEDS

Se trata de una TEDS opcional. Proporciona una serie de constantes que pueden ser usadas para describir la función de transferencia del transductor. Los factores que afectan a la función de transferencia son el sensor, el acondicionamiento de señal analógica, el filtro de anti-aliasing, y el tratamiento de señal digital. La función de transferencia da la respuesta punto a punto desde sensor analógico hasta la salida digital. Es requerido para permitir al NCAP u otro elemento en el sistema compensar la respuesta frecuencial del transductor.

TransferFunction TEDS es normalizada a la frecuencia de referencia, es decir, se asume que los datos de transductor son referidos a esta frecuencia.

6.7.10. Text-based TEDS

Es una clase de TEDS opcional.

Text-Based TEDS son una familia de TEDS que proporcionan información en texto de un TIM o TransducerChannel. Estas TEDS pueden estar en uno o más lenguajes (xml, text, html). Se componen de un directorio que facilita el acceso a un sub-bloque de lenguaje particular dentro de la TEDS seguido por los bloques de TEDS que se definen utilizando XML. La función de esta TEDS es proporcionar la información para mostrar (display) un operador. Seis TEDS son catalogadas así: Meta Identification TEDS, TransducerChannel Identification TEDS, Calibration-Identification TEDS, Commands TEDS and the Location and Title TEDS y Geographic Location TEDS.

6.7.11. End user application specific TEDS

Se trata de una TEDS opcional que proporciona la posibilidad de almacenar datos de aplicaciones que el usuario desee guardar con el TIM o TransduccerChannel. El usuario determinará el contenido y la función del End User Application Specific TEDS. Además este tipo de TEDS debe de ser capaz de ser leído y escrito.

6.7.12. Manufacturer-defined TEDS

Manufacturer-defined TEDS puede ser requerido en cualquier formato del software de la aplicación del fabricante. Un sistema normal no intentará analizar estas TEDS o al menos, interpretar su contenido de una determinada manera.

7. IEEE 1451.5

7.1.Introducción

IEEE-1451.5, aprobado en 2007, define o establece el estándar para una comunicación inalámbrica y el formato de la información para el transductor. También define el formato electrónico de las especificaciones del transductor TEDS (PHY-TEDS).

NCAP—Este dispositivo conecta uno o más WTIMs a trav´s del módulo de comunicaciones de IEEE 1451.5 y debe de estar conectado a una red.

Este estándar introduce el concepto de un Módulo Interfaz Transductor Inalámbrico (WTIM), conectado inalámbricamente sobre un NCAP, es decir, se sustituye el anteriormente mencionado TIM por otro inalámbrico llamado WTIM. A continuación se concretan algunas funciones de los dispositivos en 1451.5, se puede observar que la mayoría de estas funcionalidades son las mismas que las de un TIM:

- Un NCAP puede tener registrados múltiples WTIMs.
- Un NCAP puede dirigir datos e instrucciones desde una red externa hasta un actuador que esté conectado a un WTIM.
- Un WTIM se registra con un único NCAP.
- Un WTIM puede tener conectados varios transductores.
- La comunicación WTIM-WTIM está permitida.

7.1.1. CONFIGURACIONES DE LOS DISPOSITIVOS EN IEEE 1451.5

Las siguientes 2 figuras nos muestran las configuraciones típicas permitidas de los dispositivos para los cuales el estándar IEEE1451.5 es aplicable.

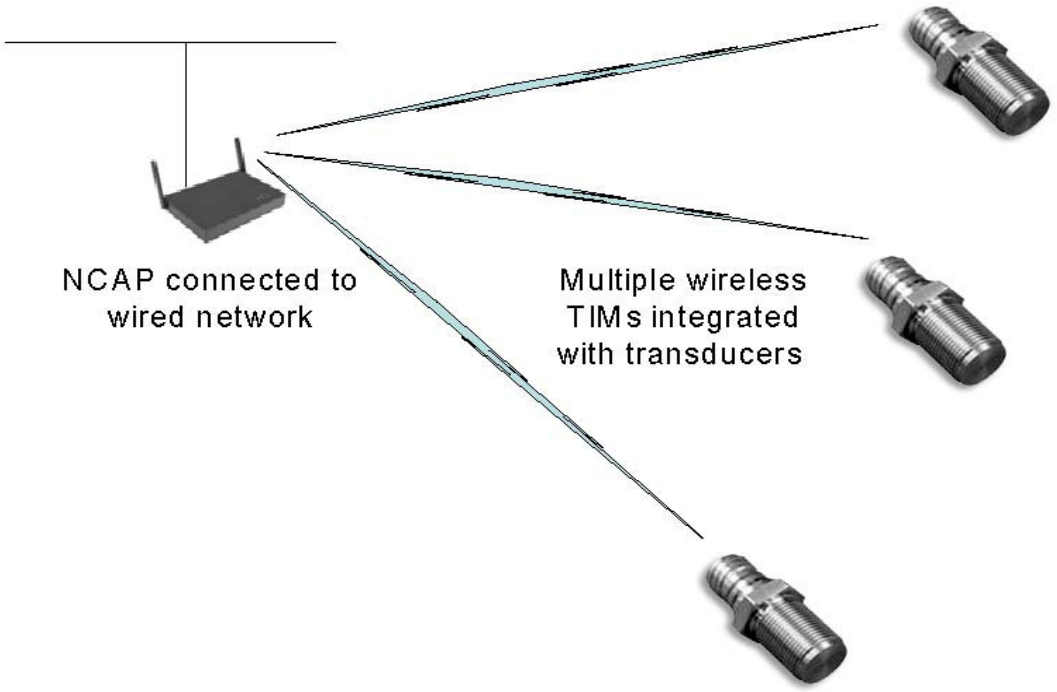


Ilustración 12: Configuración 1 permitida por EIEE 1451.5 de los dispositivos NCAP y WTIM.

La primera figura ilustra como el transductor (sensor o actuador) o transductores están integrados junto con el WTIM formando parte del mismo dispositivo. Este dispositivo será más grande que un sensor pero tiene comunicación directa con su NCAP correspondiente.

Este tipo de configuración es ideal para domicilios aislados o cuando se necesite una red de dimensiones reducidas. En cambio no es indicado para edificios,

residencias o para cuando se necesite que el sensor sea lo más pequeño posible por algún motivo.

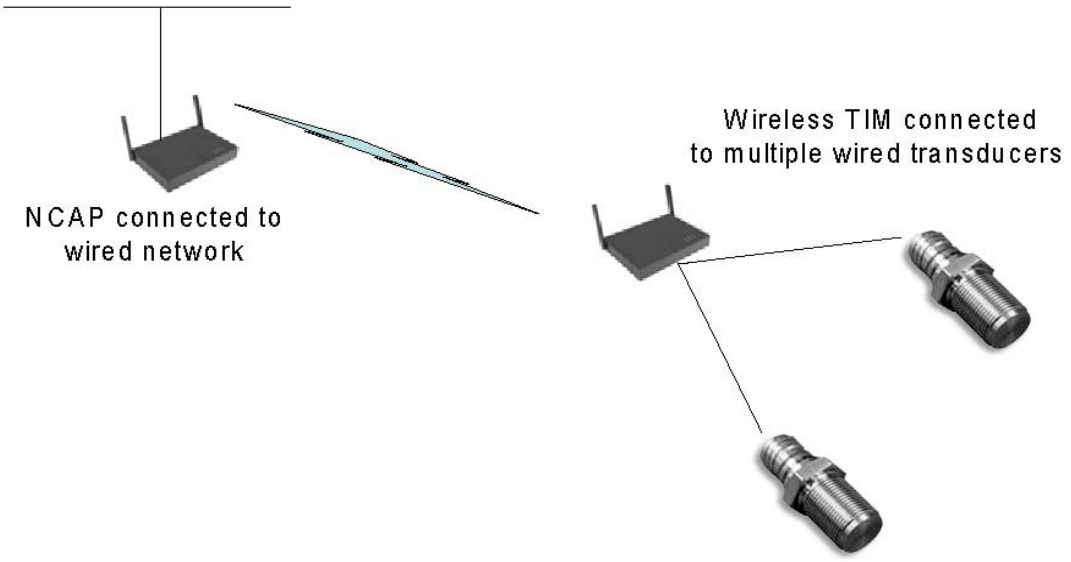


Ilustración 13: Configuración 2 permitida por IEEE 1451.5 de los dispositivos NCAP y WTIM.

En esta segunda configuración se puede observar como el WTIM es ahora un dispositivo específico y aparte del resto, sigue manteniendo la comunicación inalámbrica con el NCAP pero en este caso la comunicación entre el WTIM y los transductores (sensor o actuador) se realiza a través de cable.

Un inconveniente de esta configuración es que vuelve a aparecer el cable como elemento de comunicación y cuando se necesite colocar un transductor a una cierta distancia de un WTIM, se tendría que cablear esa distancia o colocar un WTIM cerca del transductor.

Por otra parte, existe otra posibilidad de configuración de los dispositivos, la cual **no está contemplada por el estándar IEEE 1451.5** pero puede llegar a ser también muy práctica o cuando menos, interesante. Se trata de tener un WTIM como un dispositivo aparte de los transductores y que tengan comunicación

inalámbrica con los transductores (sensores, actuadores). Al fin y al cabo esta configuración es como la anterior pero permitiendo que la comunicación transductor-WTIM sea inalámbrica en vez de por cable.

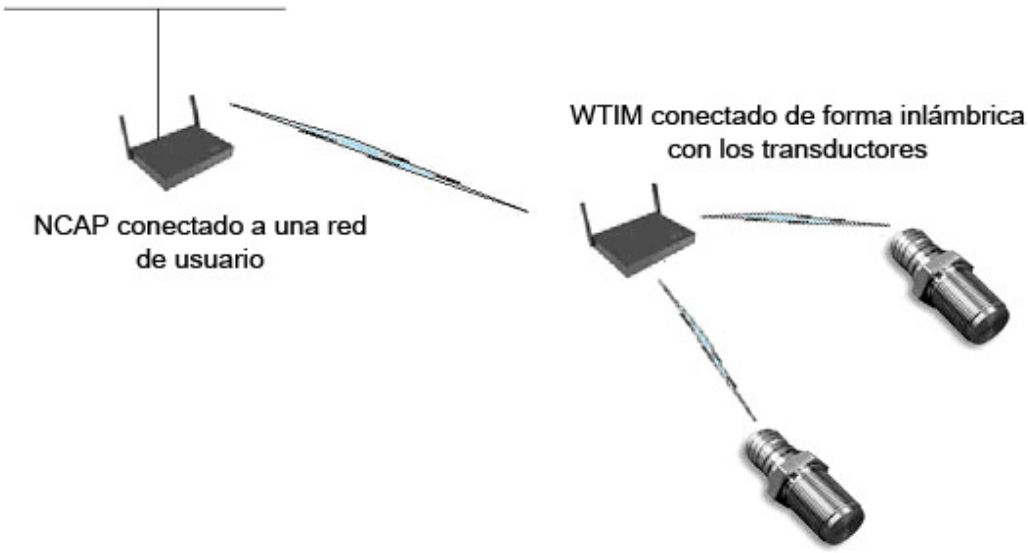


Ilustración 14: Configuración fuera de la norma IEEE 1451.5 de los dispositivos NCAP y WTIM.

En la comunicación entre el transductor y el WTIM se aprovecha el nivel de transporte ofrecido por la tecnología en uso (por ejemplo Zigbee). Esta configuración viene motivada debido a que si se dispone de un dispositivo hardware que tenga unas limitaciones en cuanto a memoria (como ocurre en ez430-RF2480 de TI, por ejemplo) es más fácil esta configuración, haciendo que el transductor envíe el dato a otro dispositivo más potente y que haga las veces de WTIM. No obstante, tener muy claro que **no** está contemplado por IEEE 1451.

IEEE 1451.5 (WirelessCommunication Protocols and TEDS Formats), establece las comunicaciones inalámbricas y protocolos que son permitidas en IEEE 1451. Los protocolos son los siguientes:

- 6LoWPAN
- IEEE 802.11

- Bluetooh
- Zigbee

7.1.2. 6LoWPAN

La subespicificación **6LoWPAN** es un acrónimo de *IPv6 over Low power Wireless Personal Area Networks*.6lowpan es el nombre de un grupo de trabajo en el área de Internet de la IETF. El grupo ha definido la encapsulación 6lowpan y los mecanismos que permiten la compresión de cabecera que permiten a los paquetes IPv6 enviarlos y (recibirlos) desde redes que están basadas en IEEE 802.15.4.

Las IEEE 802.15.4 se caracterizan por tener capacidades más limitadas que otras WPAN o WLAN: Pequeño tamaño de trama, bajo ancho de banda, baja potencia de transmisión. Se utiliza en numerosas aplicaciones embebidas y por lo general requieren comunicación multi-hop entre numerosos nodos de bajo coste para poder cubrir mayor área.

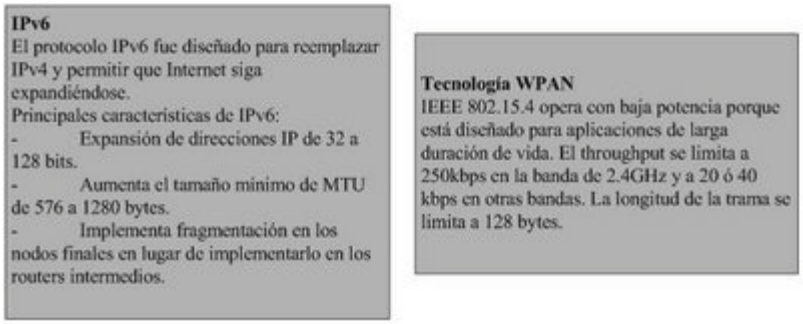


Ilustración 15: 6LoWPAN.

Suponiendo que IP requería demasiada memoria y ancho de banda, muchos han adaptado protocolos propietarios como por ejemplo Zigbee, (como se verá un poco más adelante). Teniendo presentes las restricciones de IPv6 y de las tecnologías LoWPAN (low -power wireless personal area network), soportar IPv6 sobre LoWPAN implica varios retos. Por un lado los datagramas IPv6 no son una partición natural para LoWPAN lo que implica que para poder operar

eficientemente es necesaria una fragmentación y compresión de los datagramas.

El IETF 6LoWPAN working group ha documentado todas las características en el RFC 4944.6LoWPAN introduce una capa de adaptación, entre la capa de enlace y la de red, que permite realizar comunicaciones eficientes IPv6 sobre enlaces IEEE 802.15.4 LoWPAN. El formato define como llevar a cabo comunicaciones IPv6 sobre tramas 802.15.4 y especifica los elementos clave de la capa de adaptación.

Los 3 elementos principales de 6LoWPAN:

- Compresión de cabecera
- Fragmentación
- Reenvío de datagramas IPv6 sobre capa 2

7.1.3. WI-FI 802.11

Otro protocolo inalámbrico contemplado por IEEE 1451.5 es **IEEE 802.11** o Wi-Fi de IEEE.

La especificación IEEE 802.11 (ISO/IEC 8802-11) es un estándar internacional que define las características de una red de área local inalámbrica (WLAN). **Wi-Fi** (que significa "Fidelidad inalámbrica", a veces incorrectamente abreviado WiFi) es el nombre de la certificación otorgada por la Wi-Fi Alliance, anteriormente WECA (Wireless Ethernet Compatibility Alliance), grupo que garantiza la compatibilidad entre dispositivos que utilizan el estándar 802.11. Por el uso indebido de los términos (y por razones de marketing) el nombre del estándar se confunde con el nombre de la certificación. Una red Wi-Fi es en realidad una red que cumple con el estándar 802.11. A los dispositivos certificados por la Wi-Fi Alliance se les permite usar este logotipo:

Con Wi-Fi se pueden crear redes de área local inalámbricas de alta velocidad siempre y cuando el equipo que se vaya a conectar no esté muy alejado del punto de acceso. En la práctica, Wi-Fi admite ordenadores portátiles, equipos de escritorio, asistentes digitales personales (PDA) o cualquier otro tipo de dispositivo de alta velocidad con propiedades de conexión también de alta velocidad (11 Mbps o superior) dentro de un radio de varias docenas de metros en ambientes cerrados (de 20 a 50 metros en general) o dentro de un radio de cientos de metros al aire libre.

Los proveedores de Wi-Fi están comenzando a cubrir áreas con una gran concentración de usuarios (como estaciones de trenes, aeropuertos y hoteles) con redes inalámbricas. Estas áreas se denominan "zonas locales de cobertura".

El estándar 802.11 establece los niveles inferiores del modelo OSI para las conexiones inalámbricas que utilizan ondas electromagnéticas, por ejemplo:

- La capa física (a veces abreviada capa "PHY") ofrece tres tipos de codificación de información.
- La capa de enlace de datos compuesta por dos subcapas: control de enlace lógico (LLC) y control de acceso al medio (MAC).

La capa física define la modulación de las ondas de radio y las características de señalización para la transmisión de datos mientras que la capa de enlace de datos define la interfaz entre el bus del equipo y la capa física, en particular un método de acceso parecido al utilizado en el estándar Ethernet, y las reglas para la comunicación entre las estaciones de la red. En realidad, el estándar 802.11 tiene tres capas físicas que establecen modos de transmisión alternativos:

Capa de enlace de datos (MAC)	802.2
	802.11
Capa física (PHY)	DSSS | FHSS | Infrarrojo

Ilustración 16: Capas WI-FI.

Cualquier protocolo de nivel superior puede utilizarse en una red inalámbrica Wi-Fi de la misma manera que puede utilizarse en una red Ethernet.

El estándar 802.11 en realidad es el primer estándar y permite un ancho de banda de 1 a 2 Mbps. El estándar original se ha modificado para optimizar el ancho de banda (incluidos los estándares 802.11a, 802.11b y 802.11g, denominados estándares físicos 802.11) o para especificar componentes de mejor manera con el fin de garantizar mayor seguridad o compatibilidad. De entre todos los 802.11x existentes, el estándar 802.11b es el más utilizado actualmente. Ofrece un rendimiento total máximo de 11 Mbps (6 Mbps en la práctica) y tiene un alcance de hasta 300 metros en un espacio abierto. Utiliza el rango de frecuencia de 2,4 GHz con tres canales de radio disponibles.

7.1.4. Bluetooth

La subespecificación de **Bluetooth** para sistemas IEEE 1451.5 define las funciones, interfaces y protocolos que serán activas mediante una comunicación inalámbrica entre un WTIM y un NCAP. Esta especificación apoyará el Bluetooth 2.0 + EDR o especificaciones posteriores con el objetivo de especificar las funciones, el interfaz, y protocolos para proporcionar la tecnología Bluetooth inalámbrica a este estándar.

El estándar Bluetooth, del mismo modo que Wi-Fi, utiliza la técnica FHSS (*Frequency Hopping Spread Spectrum*, en español *Espectro ensanchado por saltos*

de frecuencia), que consiste en dividir la banda de frecuencia de 2.402 - 2.480 GHz en 79 canales (denominados *saltos*) de 1 MHz de ancho cada uno y, después, transmitir la señal utilizando una secuencia de canales que sea conocida tanto para la estación emisora como para la receptora.

Por lo tanto, al cambiar de canales con una frecuencia de 1600 veces por segundo, el estándar Bluetooth puede evitar la interferencia con otras señales de radio.

El estándar Bluetooth se basa en el modo de operación maestro/esclavo. El término "**piconet**" se utiliza para hacer referencia a la red formada por un dispositivo y todos los dispositivos que se encuentran dentro de su rango. Pueden coexistir hasta 10 piconets dentro de una sola área de cobertura. Un dispositivo maestro se puede conectar simultáneamente con hasta 7 dispositivos esclavos activos (255 cuando se encuentran en modo *en espera*). Los dispositivos en una piconet poseen una dirección lógica de 3 bits, para un máximo de 8 dispositivos. Los dispositivos que se encuentran en el modo *en espera* se sincronizan, pero no tienen su propia dirección física en la piconet.

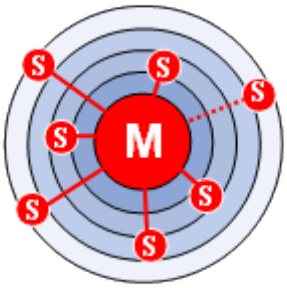


Ilustración 17: Conexión de dispositivos esclavos activos bluetooth.

En realidad, en un momento determinado, el dispositivo maestro sólo puede conectarse con un solo esclavo al mismo tiempo. Por lo tanto, rápidamente cambia de esclavos para que parezca que se está conectando simultáneamente con todos los dispositivos esclavos.

Bluetooth permite que dos piconets puedan conectarse entre sí para formar una red más amplia, denominada "**scatternet**", al utilizar ciertos dispositivos que actúan como puente entre las dos piconets.

El establecimiento de una conexión entre dos dispositivos Bluetooth sigue un procedimiento relativamente complicado para garantizar un cierto grado de seguridad, como el siguiente:

- Modo pasivo
- Solicitud: Búsqueda de puntos de acceso
- Paginación: Sincronización con los puntos de acceso
- Descubrimiento del servicio del punto de acceso
- Creación de un canal con el punto de acceso
- Emparejamiento mediante el PIN (seguridad)
- Utilización de la red

7.1.5. Zigbee

En último lugar, la subespecificación **Zigbee** que, de la misma forma que las anteriores, va a proporcionar la capa de transporte para una comunicación inalámbrica entre los WTIM y NCAP. Las características que lo diferencian de otras tecnologías son su bajo consumo, su topología de red en malla y su fácil integración (se pueden fabricar nodos con muy poca electrónica).

Zigbee es un estándar de comunicaciones inalámbricas diseñado por la Zigbee Alliance. Es un conjunto estandarizado de soluciones que pueden ser implementadas por cualquier fabricante. Zigbee está basado en el estándar IEEE 802.15.4 (de la misma forma que 6LoWPAN que se ha visto anteriormente) de redes inalámbricas de área personal (wireless personal area Network, WPAN) y tiene como objetivo las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías.

ZigBee es un sistema ideal para redes domóticas, específicamente diseñado para reemplazar la proliferación de sensores/actuadores individuales. ZigBee fue creado para cubrir la necesidad del mercado de un sistema a bajo coste, un estándar para redes wireless de pequeños paquetes de información, bajo consumo, seguro y fiable.

Las tecnologías inalámbricas han adoptado con el paso del tiempo una manera más sencilla y cómoda de utilizar toda clase de dispositivos con el fin de mejorar el confort y las comunicaciones en general. Ésta investigación aborda la tecnología inalámbrica ZigBee, basada en el estándar 802.15.4 que por su poca introducción al mercado no es muy conocida a pesar de que no es muy reciente.

ZigBee comunica una serie de dispositivos haciendo que trabajen más eficiente entre sí. Es un transmisor y un receptor que usa baja potencia para trabajar y tiene como objetivo las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías. Es ideal para conexiones con diversos tipos de topología, lo que a su vez lo hace más seguro, barato y que no haya ninguna dificultad a la hora de su construcción porque es muy sencilla.

En el siguiente apartado, se entra a fondo en la introducción de Zigbee en el estándar IEEE 1451 y de cómo se crea la capa de convergencia del estándar y de Zigbee.

7.2.Zigbee en IEEE 1451

7.2.1. Introducción

La sub-especificación IEEE 1451.5 para ZigBee se refiere exclusivamente a los requisitos necesarios para permitir el uso de redes ZigBee para proporcionar una capa de transporte para la aplicación del estándar IEEE 1451.5.

La especificación define los perfiles de aplicación ZigBee, aspectos de seguridad y funcionalidad de la calidad del servicio (QoS), y define una capa de convergencia para la interfaz entre la red ZigBee y IEEE 1451.0.

El protocolo de red ZigBee lo define ZigBee Alliance (www.zigbee.org). Todas las referencias a ZigBee se refieren a las implementaciones de la versión del protocolo ZigBee 1.0 tal como se define por ZigBee Alliance.

A continuación se muestra el *stack* de Zigbee en 1451.5:

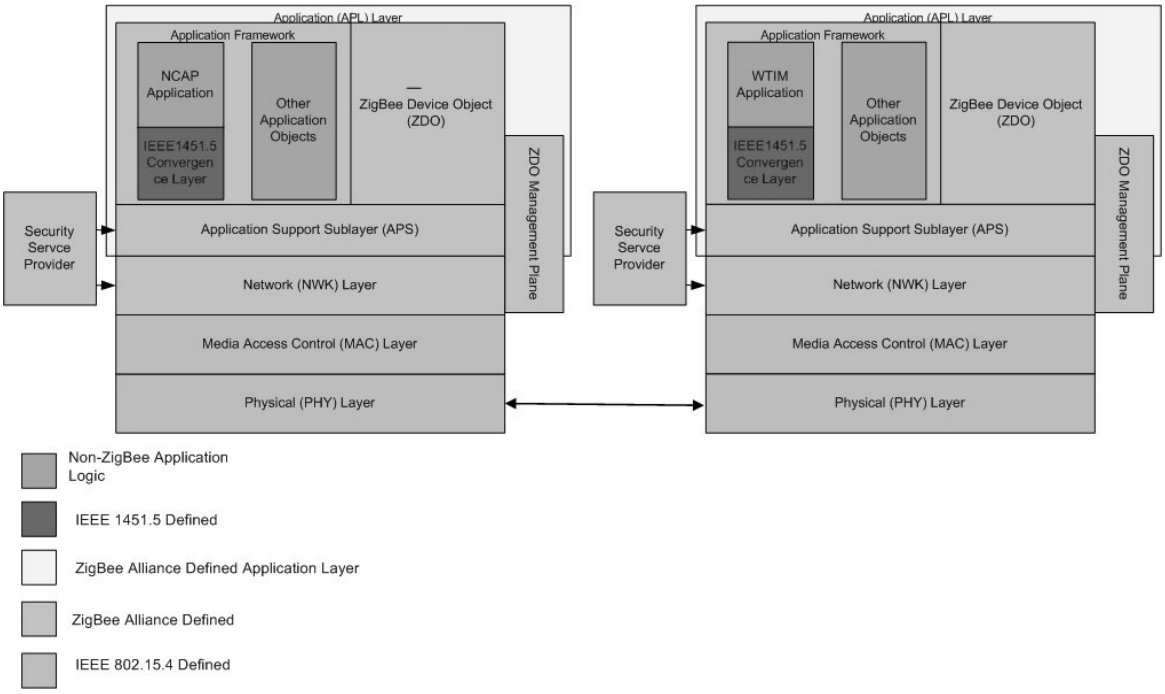


Ilustración 18: Arquitectura de Zigbee/802.15.4 en IEEE 1451.5.

La capa física (PHY) y la capa de control de acceso medio (MAC) se definen por el IEEE Std. 802.15.4-2003. El protocolo ZigBee se basa en estas capas y proporciona las definiciones de una capa de red (NWK) y de capa de aplicación (APL).

La funcionalidad que ofrece IEEE 1451.5 se integra dentro de la APL de ZigBee.

Una red ZigBee requiere la presencia de un coordinador ZigBee así como los dispositivos de función completa y de función reducida (FFD y RFD). El FFD puede actuar como un router ZigBee, mientras que el dispositivo RFD no puede enrutar el tráfico y está restringido a actuar como nodo final, llamado *endpoint*.

Un NCAP debe ser capaz de funcionar, como mínimo, como un RFD (si no realiza el enrutamiento del tráfico de ZigBee)

Los WTIMs no tienen por qué conectarse directamente con un NCAP, tampoco se requiere que entre ellos tengan una relación padre/hijo. La capa de convergencia ZigBee mapea los DestinationIds (DestIds) de IEEE 1451.0 con las direcciones de la red ZigBee permitiendo a la red ZigBee enrutar el tráfico entre WTIMS y su NCAP asociado, de una forma transparente.

7.2.1.1.Perfiles de aplicación, endpoints y clusters. Introducción a Zigbee

Para comprender mejor la utilización que hace IEEE 1451.5 sobre Zigbee se va a explicar a continuación aspectos básicos de Zigbee que son necesarios y básicos para IEEE 1451.5 como son los perfiles de aplicación, endpoints y clusters.

Los perfiles de aplicación y clusters se usan en la especificación ZigBee para definir la funcionalidad de dispositivos. Esta funcionalidad es proporcionada por un dispositivo específico así como los grupos relacionados de estos.

Los perfiles son las colecciones de clusters que juntos proporcionan el marco necesario para comunicarse e interactuar con los dispositivos.

ENDPOINTS

En un nodo ZigBee encontramos los endpoints. Los endpoints, identificados por un número entre 1 y 240, son objetos de aplicación que definen cada una de las

aplicaciones que soporta un nodo. Los endpoints sirven para dos propósitos en ZigBee:

- Permiten que en un nodo existan diferentes perfiles de aplicación.
- Permiten que en un nodo existan diferentes dispositivos.

Podemos imaginar un cable virtual conectando un endpoint en un nodo con un endpoint de otro nodo (figura 1). En la figura 1-a) el interruptor sólo tiene un endpoint que se comunica con tres endpoints de tres bombillas y, por tanto, controla a las bombillas como si fueran una, mientras que el interruptor de la figura 1-b) tiene tres endpoints, uno por bombilla y, de esa forma, puede controlar a cada una de manera independiente.

Cada petición de datos ZigBee es enviada (y recibida) sobre un Perfil de Aplicación ('Application Profile'). Los ID de los Perfiles de Aplicación son números de 16-bit:

- 0x0000 – 0x7FFF: Perfiles Públicos
- 0xBF00 – 0xFFFF: Perfiles Privados

PERFILES DE APLICACIÓN

Podemos pensar en un perfil como un espacio de aplicaciones y dispositivos relacionados. Los perfiles públicos son aquellos especificados por la Alianza ZigBee (en contraposición a los perfiles privados que están definidos por fabricantes particulares).

Por ejemplo "*Home Automation*" es un perfil de aplicación público que define un amplio conjunto de dispositivos destinados a la monitorización del hogar, incluyendo iluminación, control remoto y sistemas de climatización, entre otros. La siguiente tabla muestra los perfiles públicos que ha implementado la *Zigbee Alliance*.

Profile Name
Industrial Plant Monitoring
Home Automation
Commercial Building Automation
Telecom Applications
Health Care
Advanced Metering Initiative
Remote Control
Smart Energy

Tabla 18: Identificadores de los perfiles de Zigbee.

En una red ZigBee pueden coexistir diferentes perfiles de aplicación, públicos y privados. Los perfiles públicos están diseñados de tal forma que los productos de un fabricante puedan trabajar 'out-of-the-box' con productos de otro fabricante. Por ejemplo, una bombilla Philips podría trabajar con un interruptor Leviton.

CLUSTERS

Los *clusters*, definidos por un identificador de 16-bit son objetos de aplicación. Así como la NwkAddr y el *endpoint* (no entraremos a definir esto) son conceptos de direccionamiento en una red ZigBee, el cluster tiene que ver con el significado de la aplicación. En la siguiente tabla incluimos varios clusters extraídos de la Librería de Clusters ZigBee (ZCL, ZigBee Cluster Library).

Cluster Name	Cluster ID
Basic Cluster	0x0000
Power Configuration Cluster	0x0001
Temperature Configuration Cluster	0x0002
Identify Cluster	0x0003
Groups Cluster	0x0004
Scenes Cluster	0x0005
OnOff Cluster	0x0006
OnOff Configuration Cluster	0x0007
Level Control Cluster	0x0008
Time Cluster	0x000a
Location Cluster	0x000b

Tabla 19: Identificadores de los Cluster Zigbee.

Supongamos un cluster en particular, el *OnOff Cluster (ID 0x0006).* Este cluster sabe cómo hacer que "algo" cambie su estado de ON a OFF y viceversa. No importa que ese "algo" sea una bombilla, un timbre o un mecanismo para abrir y cerrar una puerta. La única condición es que el dispositivo en cuestión tenga un estado binario: on/off; open/close.

Los clusters sólo tienen sentido en el contexto de un perfil de aplicación. Por ejemplo, un perfil privado puede definir el cluster 0x0000 como el cluster "apagado de la red", mientras que en el perfil público Home Automation el cluster 0x0000 es el Basic Cluster. Todos los perfiles públicos soportan la ZigBee Cluster Library, de tal forma que utilizan los mismos cluster IDs y los productos de diferentes fabricantes pueden comunicarse correctamente. Los perfiles privados no tienen por qué ajustarse a los clusters de la ZCL, pudiendo definir los suyos propios.

Los clusters, además del identificador, tienen sentido. Existen clusters de salida y clusters de entrada. ZigBee establece esta diferenciación para realizar correctamente la unión virtual de dos endpoint de dos nodos. Para realizar esta unión de dos endpoint, ambos deben incluir en su descriptor simple el mismo cluster pero de sentido contrario, es decir, en uno de ellos el cluster será de entrada y en el otro el cluster será de salida.

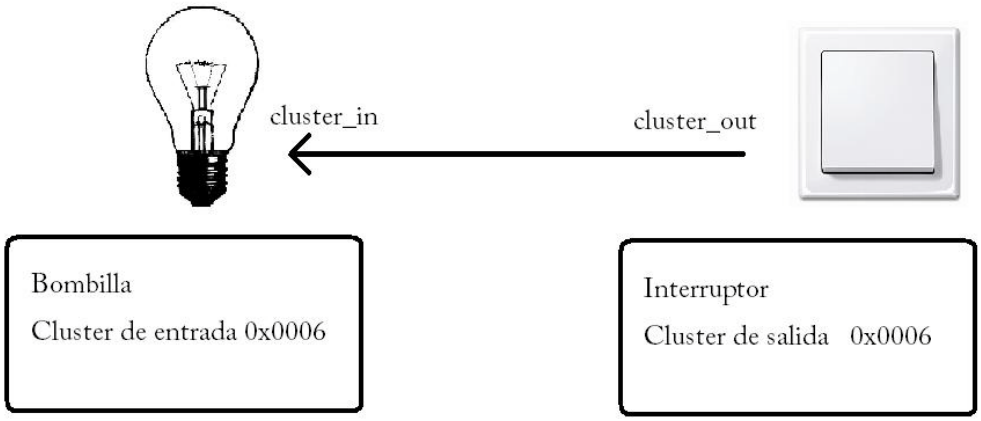


Ilustración 19: Ejemplo de aplicación sencilla de los Cluster.

Vemos que están asociados el interruptor con la bombilla y que la flecha que los une va del interruptor a la bombilla. Esto es así porque el interruptor puede actuar sobre la bombilla, mientras que la bombilla sólo recibe órdenes. Por tanto, el interruptor tiene el cluster 0x0006 y su sentido es de salida (envía órdenes) y la bombilla tiene el mismo cluster 0x0006 pero su sentido es de entrada (recibe órdenes).

Dentro de los clusters encontramos comandos y atributos. Una aplicación ZigBee puede averiguar si una bombilla está encendida o apagada gracias al atributo OnOff del cluster OnOff, o puede causar una acción sobre el estado de esa misma bombilla utilizando los comandos on, off o 'toggle' (cambio de estado) del cluster OnOff.

La subespecificación de Zigbee en IEEE 1451.5 define un sólo perfil de aplicación ZigBee que contiene dos clusters que proporcionarán la funcionalidad necesaria de usar una red de ZigBee como transporte para IEEE 1451.5 comunicando así el NCAP con WTIM.

Todas los estándares ZigBee aplicables, perfiles, y clusters son asumidos para ser usados junto con ZigBee CII (Comercial, Industrial, e Institucional).

7.2.2. Características de la Capa Aplicación

Esta tabla describe las características necesarias para unidades que usan ZigBee IEEE 1451.5 BULK TRANSFER PROFILE.

	Característica	En WTIM	En NCAP
1.	NCAP Service Interface	No Aplicable	Obligatorio
2.	WTIM Service Interface	Obligatorio	No Aplicable
3.	TEDS	Obligatorio	No Aplicable
4.	WTIM Reconnect	Opcional	Obligatorio
5.	MIB	No Aplicable	Opcional

Tabla 20: ZigBee IEEE 1451.5 Bulk Transfer Profile overview.

7.2.3. Funciones del perfil de aplicación

Este perfil de aplicación se apoya en "*Zigbee Message Format*", un formato de mensaje descrito a continuación.

Command Type *1	Packet ID *2	Sequence Number	Octet Data *3

Tabla 21: Formato de mensaje Zigbee en IEE1451.5.

NOTA *1*—CommandType es un valor sin signo de 8 bit, incluyendo 4 bits para indicaciones de estado y 4 bits para el tipo de comando. Los campos de estado están en los cuatro bits más significativos, y el tipo de comando es almacenado en los cuatro bits menos significativos. Sólo dos valores de CommandType son apoyados: SET y SET RESPONSE con los valores de 0x01 y 0x09, respectivamente. Sólo un bit del campo de estado es definido; el bit más significativo debería ser interpretado como el "MoreBit" descrito como en la NOTA 2.

NOTA *2*—Se aumenta el valor que comienza en 0. Cuando el MoreBit es puesto a "1" (el bit más significativo del campo de CommandType), indica que no es el último fragmento en el paquete y el receptor debería esperar fragmentos adicionales hasta que un mensaje con el MoreBit a "0" sea recibido indicando que este es el último fragmento. Si el cambio de valor PacketID es antes de que se hayan recibido todos los fragmentos, el receptor debe tomar esto como una indicación de que los fragmentos se han perdido o retrasado y puede optar por ignorar y desechar cualquier paque te recibido parcial antes de este tiempo, o optar por conservar los paquetes antes de parciales de hasta a 5 s en un intento de dar cabida a fragmentos retrasados o fuera de tiempo. Si se encuentra en alguno algún error, emitirá una SET-RESPONSE al remitente con el código de error apropiado.

NOTA *3*—La longitud de la cadena del octeto es codificada en el primer octeto. Para los comandos de SET, el campo de OctetData contendrá los datos siendo transferidos; para SET-RESPONSE, el campo de OctetData contendrá un código de estado definido como el uno o el otro 0x00 para el Éxito y 0x01 para cualquier error (estos valores es enviado como cuerdas de octeto y es precedido con un valor de longitud de cadena de octeto de 0x01).

Cada mensaje que se envíe se confirma por un mensaje *Set-Response* con un código de estado contenido en el campo de *OctetData*, informando sobre la recepción completa del mensaje o la detección de un error. En el campo del *PacketID* se pone el valor de identificación del paquete correspondiendo al paquete confirmado (acknowledged).

Indicación del Estado	Código
Succes	0x00
Error- Lost Fragment	0x80
Error- Malformed Message	0x81
Error- Other Error	0xFF

Tabla 22: Identificadores de los Estados de confirmación de los mensajes.

El perfil de aplicación descrito por IEEE 1451.5 (subespicificación Zigbee) apoya dos dispositivos: **NCAP y WTIM.**

Para los objetivos de esta especificación, este perfil ZigBee usará el Identificador de Perfil ZigBee Privado 0xBF00, asignado por la Alianza ZigBee.

7.2.3.1.Los clusters

La siguiente figura proporciona una representación gráfica del modelo de cluster definido por este perfil de aplicación. La figura muestra todos los clusters de entrada y de salida así como cualquier relación definida por este perfil de aplicación. Este perfil de aplicación utiliza *ZigBee IEEE 1451.5 Bulk Transfer Profile cluster model.*

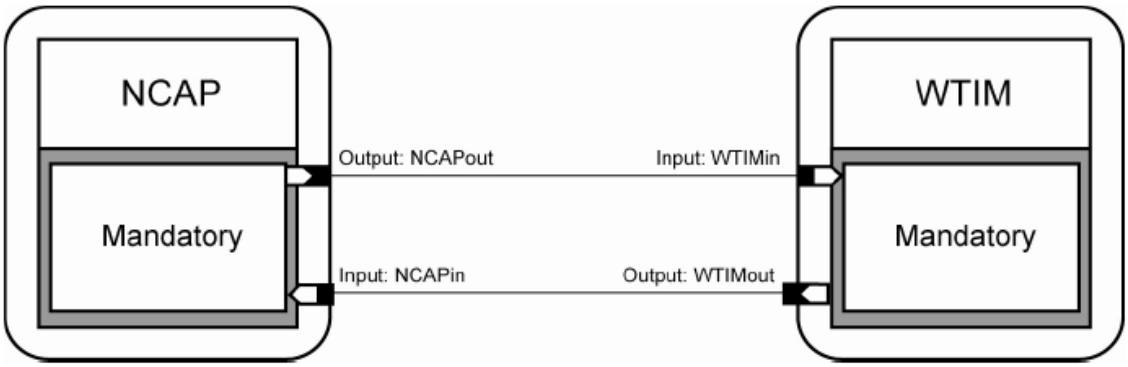


Ilustración 20: Clusters obligatorios para Zigbee en IEEE 1451.5.

Dentro de este perfil de aplicación, los clusters obligatorios y todos sus atributos definidos en las descripciones del dispositivo, son comunes a todo el IEEE 1451.5 (subespecificación ZigBee) y por lo tanto son usados por todos los dispositivos que emplean este perfil de aplicación y todas sus revisiones. Si hubiera alguna variación se indicaría.

ClusterNames de la figura anterior son predefinidos como *Entrada*: o *Salida*: de forma que los modelos que se pongan en práctica son fáciles de implementar ya que los dispositivos contienen las mismas descripciones. Sólo los dispositivos de entrada s unen con el dispositivo de salida que tenga su mismo ClusterName (NCAP con WTIM y WTIM con NCAP).

ClusterIDs y descripciones Cluster

Clusters Obligatorios		
Nombre Cluster	Cluster ID	Descripción
NCAPout	0x01	Cluster NCAP como emisor
NCAPin	0x02	Cluster NCAP como receptor
WTIMin	0x01	Cluster WTIM como receptor
WTIMout	0x02	Cluster WTIM como emisor
Clusters Opcionales		
Nombre Cluster	Cluster ID	Descripción
No definidos		

Tabla 23: Identificadores de los Clusters y sus Definiciones.

En las siguientes tablas se describen los clusters soportados para el NCAP y para el WTIM.

Clusters Direccionamiento Directo para NCAP		
Nombre del Cluster	Formato Mensaje	Descripción
Output:NCAPout	Zigbee IEEE 1451.5 Bulk Transfer	Bulk transfer originado desde un dispositivo NCAP
Input:NCAPin	Zigbee IEEE 1451.5 Bulk Transfer	Bulk transfer que llega a un dispositivo NCAP

Tabla 24:Direccionamiento de los Clusters para un NCAP.

Clusters Direccionamiento Directo para WTIM		
Nombre del Cluster	Formato Mensaje	Descripción
Output:WTIMout	Zigbee IEEE 1451.5 Bulk Transfer	Bulk transfer originado desde un dispositivo WTIM
Input:WTIMin	Zigbee IEEE 1451.5 Bulk Transfer	Bulk transfer que llega a un dispositivo WTIM

Tabla 25: Direccionamiento de los Clusters para WTIM.

7.2.3.2.Escenario de uso

ZigBee IEEE 1451.5 Bulk Transfer Profile define la forma de cómo hay que configurar la conexión ZigBee desde el NCAP a un WTIM, e iniciar la transferencia de datos a/de un transductor. Normalmente esto ocurre en cinco fases:

Registro: Se encuentra (discovery) y se registra un WTIM con un NCAP.

Query TEDS: recupera la información sobre el WTIM y sus transductores conectados.

Configuración de Transductor: configurar el WTIM para cambiar datos con un transductor conectado.

Intercambio de datos: entre NCAP y WTIM de\a un transductor para realizar la medida.

7.2.3.3.Mapeo con IEEE 1451.0

En IEEE 1451.0, un NCAP y sus asociados WTIMs forman un grupo de comunicaciones lógicas. A cada nodo (NCAP o WTIM) la capa IEEE 1451.0 le asigna un identificador Id UINT16 único dentro del grupo. El valor 0x0000 está reservado como dirección de Broadcast, y las direcciones son asignadas ordenadamente comenzando por 0x0001.

Este " IEEE 1451.0 Destination ID " se denomina como "destID" en IEEE 1451.5.

IEEE 1451.0define que un WTIM sólo puede ser asociado con único NCAP. Desde la capa IEEE 1451.0 de un WTIM, el NCAP “destID” es redundante. En este caso, todas las comunicaciones de un WTIM a su NCAP asociado serán enviadas a la dirección del dispositivo ZigBee correspondiente. Es importante notar que el

término "asociación" se refiere tan sólo a la asociación lógica IEEE 1451.0 y no dice nada sobre una relación padre/hijo entre dos nodos Zigbee.

Los WTIMs son descubiertos por la capa IEEE 1451.5 del NCAP, pero es la capa IEEE 1451.0 del NCAP la encargada de llevar a cabo el registro. El IEEE 1451.0 le asignará el *destId*, y la capa IEEE 1451.5 almacenará en caché la dirección Zigbee (ZD_ADDR) que corresponda y asociarlo con este *destId*.

IEEE 1451.0. Octetarray payloads

Para reducir al mínimo el conocimiento que la capa IEEE 1451.5 tiene que tener de toda la información transmitida por la capa IEEE 1451.0, se juntan en un *payload* codificado como una *OctetArray*. Salvo en los casos en que la capa IEEE 1451.5 desee interceptar un mensaje de IEEE 1451.0, la capa IEEE 1451.5 debe transportar el *payload* sin tener que analizarlo. La capa IEEE 1451.5 espera a que dicho *OctetArray* se encapsule (o empaquete) en una serie de paquetes apropiados de ZigBee. Además, la capa IEEE 1451.5 es responsable de segmentar el *OctetArray* en el tamaño adecuado al permitido por la red Zigbee y de volver a "montar" de nuevo el *OctetArray* cuando sea la receptora. Del mismo modo, todos los problemas con la encriptación, autenticación y compresión son responsabilidades de la capa IEEE 1451.5 y, en este caso, son controlados por la capa de red de ZigBee.

El payload de IEEE 1451.0 se encapsula en una serie de mensajes Zigbee utilizando clusters ZigBee anteriormente descritos. En la siguiente tabla se muestra la paquetización de dicho cluster.

Campo	Descripción	Tipo de Dato
Command Type	Indica si se trata de un comando Set o Set Response-el bit más significativo indica que hay más fragmentos pendientes en relación con el actual paquete-este bit será puesto a cero para el último paquete en la paquetización	8 bit unsigned integer
PacketID	Se incrementa cada vez que se encapsula un paquete nuevo	8 bit unsigned integer
SeqNum	Secuencialmente se incrementa el número de secuencia que identifica al fragmento de paquetes específicos que se envían. El bit más significativo es "1" para indicar que faltan paquetes por llegar. El último paquete es puesto a cero.	16 bit unsigned integer
PacketData	En este paquete se contiene la longitud y datos octetos. Dentro de este paquete se encapsula el mensaje generado por la capa IEEE 1451.0 (el cual se ha visto en el apartado 6.2 y 6.3 de este documento).	Byte Stream

Tabla 26: paquetización del Cluster.

Esta es la paquetización del cluster anteriormente definido, es el "formato" de mensaje que utiliza Zigbee para enviar o recibir información desde un Módulo de Comunicaciones de un NCAP al Módulo de Comunicaciones de un WTIM. Dentro de este mensaje, en el campo *PacketData* (Data Octets) se encapsula el mensaje que crea la capa IEEE 1451.0 del NCAP (si inicia este la comunicación) y que recibe la capa IEEE 1451.0 del WTIM.

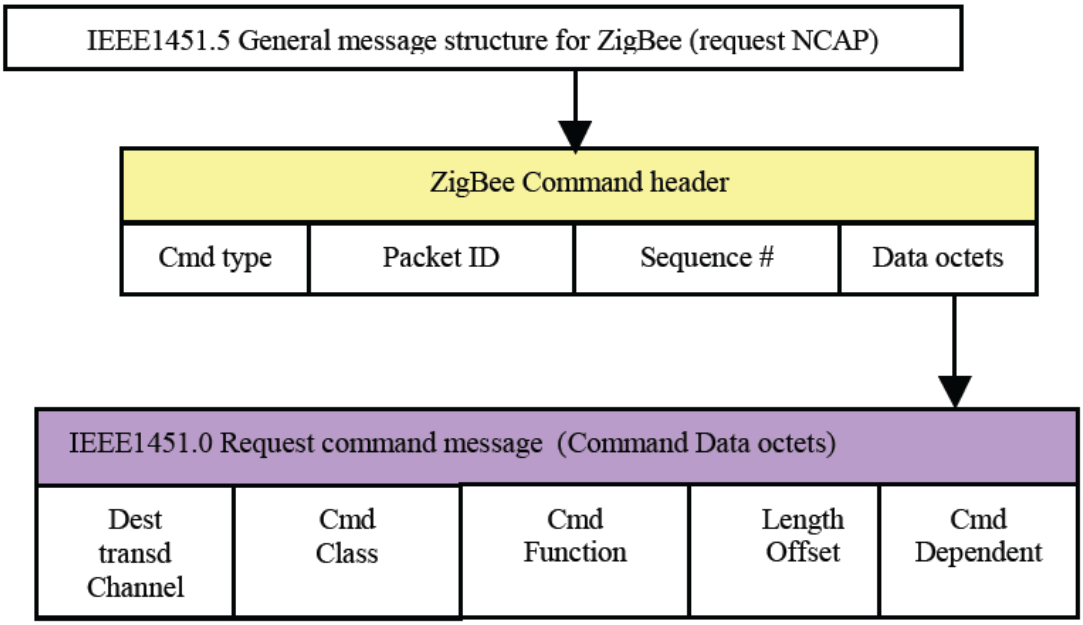


Ilustración 21: Encapsulamiento del mensaje creado en la capa IEEE 1451.0 dentro del campo DataOctects dentro del Cluster Zigbee IEEE 1451.5.

Dependiendo de ZigBee y del modo de seguridad de IEEE 802.15.4 usado, un mínimo de 44 bytes (octet datas) se pueden transferir en cada mensaje ZigBee IEEE 1451 para la transferencia de un encapsulado máximo en el mensaje IEEE 1451.0 de $2^{16} \times 44$ bytes (aproximadamente 2,88 megabytes).

CAPA DE CONVERGENCIA

Para que se produzca una comunicación satisfactoria entre un WTIM y un NCAP es necesaria de la existencia de una capa de convergencia la cual va a ser el interfaz que va a permitir el paso de 1451.5 a 1451.0.

Esta interfaz es el Módulo de Comunicaciones anteriormente comentado (apartado 6.7 de este documento). El Módulo de Comunicaciones es definido como tal en IEEE 1451.0 donde se especifican las interfaces, el nombre de las funciones y métodos que se utilizan, el tipo de variables a utilizar en cada función… Pero es en IEEE 1451.X (en nuestro caso 1451.5, subespecificación Zigbee) donde se entra a

fondo en definir el Módulo de Comunicaciones según la tecnología que utilices para comunicar el NCAP con el TIM.

Esta capa de convergencia se encuentra descrita en el apartado **8.8 de IEEE Std 1451.5**.Aquí encontraremos la guía de cómo tiene que ser el Módulo de Comunicaciones cuando deseemos utilizar tecnología Zigbee entre en NCAP y los WTIM, pero como se ha dicho se trata de un caso particular del ***Apartado 11 (Module Communications API) de IEEE Std 1451.0***.

8. NIVEL DE APLICACIÓN EN IEEE 1451

8.1.Introducción

Como se muestra en la siguiente figura, se utilizan dos interfaces de comunicación de redes para acceder a los transductores inteligentes de IEEE 1451, es decir, los protocolos de IEEE 1451.1 y el protocolo IEEE 1451.0 Hypertext Transfer Protocol (HTTP). El IEEE 1451.1 se centra principalmente en la forma cliente-servidor y protocolos publisher-subscriber. Actualmente El protocolo IEEE 1451.0 HTTP se centra principalmente en HTTP Web Access a los transductores inteligentes.

Recientemente, ha salido a la luz un conjunto de servicios Web, designado como Smart Transducer Web Services (STWS), para acceder a los transductores inteligentes de IEEE 1451.STWS se basa en Arquitectura Orientada a Servicios (SOA).

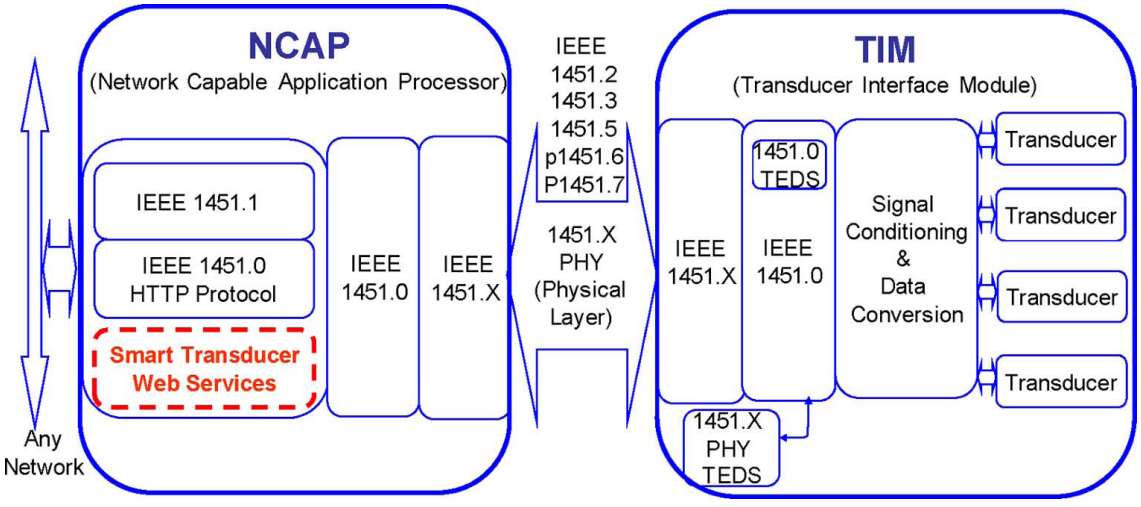


Ilustración 22: Arquirectura de las capas de IEEE 1451. Observar el nivel de aplicación de NCAP.

8.2.Smart transducer web service. STWS

STWS describe un servicio web unificado para transductores inteligentes de IEEE 1451, desarrollado por el Instituto Nacional de Estándares y Tecnología (NIST) basado en el estándar IEEE 1451.0.

Antes de comenzar a explicar la base de STWS, se va definir y explicar los fundamentos básicos con los que se construye, SOAP y SOA.

SOAP

SOAP no es más que un protocolo estándar que permite la comunicación y la interoperabilidad entre diversas aplicaciones Web desarrolladas bajo tecnologías diferentes. SOAP especifica el formato de mensaje que accede e invoca a los objetos, más que un objeto en particular. Por otra parte el término "Object" en el nombre significa que se adhiere al paradigma de la programación orientada a objetos.

Actualmente, los WS están siendo ampliamente aceptados por las empresas para el desarrollo de software de uso interno. Debido a la tecnología que es usada por los WS, y en concreto al uso de SOAP, que permite la cooperación y la interoperabilidad entre empresas que estén desarrollando proyectos en común y en las cuales no estén trabajando sobre la misma plataforma, lenguaje de programación o hardware compatibles.

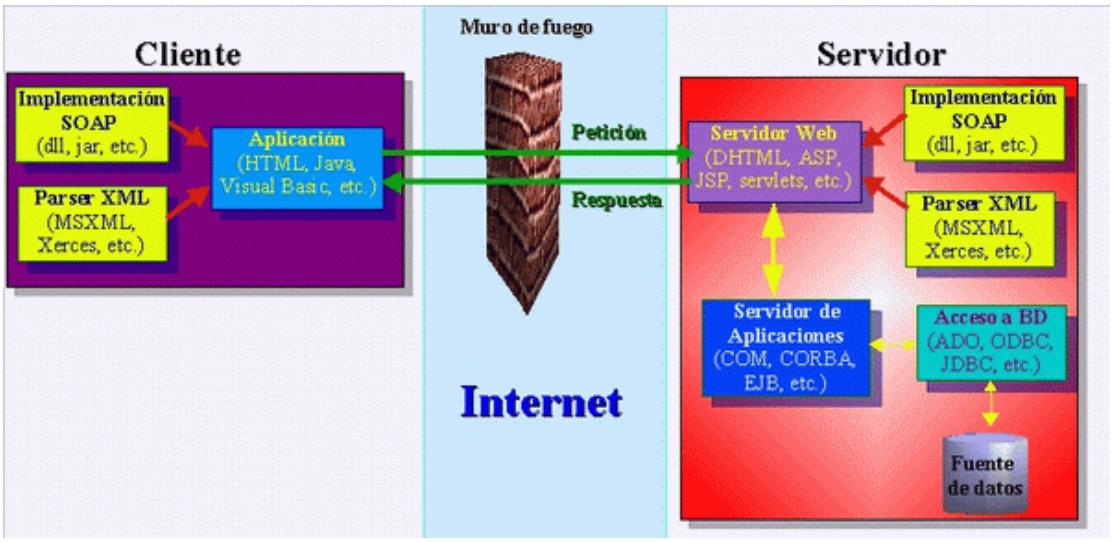


Ilustración 23: Ejemplo arquitectura SOAP.

La especificación SOAP menciona que las aplicaciones deben ser independientes del lenguaje de desarrollo, por lo que las aplicaciones cliente y servidor pueden estar escritas con HTML, DHTML, Java, Visual Basic u otras herramientas y lenguajes disponibles. Lo importante es tener alguna implementación de SOAP (dependiendo de la herramienta de desarrollo elegida) y enlazar sus librerías con la aplicación. Aunque esto no es estrictamente necesario, es preferible trabajar usando dichas librerías, con el fin de no reescribir un código ya probado.

SOA

Un aspecto muy importante sobre la Arquitectura Orientada a Servicios es su flexibilidad. Se trata esencialmente de un conjunto de servicios independientes, donde cada uno es relativamente sencillo para construirlo o reemplazarlo si es necesario. Al ser independientes, el poder unirlos permite a SOA adaptar cambios, cuestión imposible para arquitecturas tradicionales.

En la Arquitectura Orientada a Servicios, se puede reemplazar un servicio sin tener que preocuparse por la tecnología fundamental, las interfaces es lo que

importa, y está definida en un estándar universal en servicios Web y XML. Esto es flexibilidad a través de la interoperabilidad.

Ofrece también habilidad de asegurar los recursos existentes, aplicaciones y bases de datos legales y hacerlos parte de las soluciones empresariales extendiéndolos al SOA en vez de reemplazarlos. El resultado en la red es la habilidad de evolucionar rápida y eficientemente, en otras palabras, adaptarse "orgánicamente" de acuerdo a la demanda del negocio. Esto es realmente nuevo.

Cuando hablamos de servicio en SOA nos referimos a la manera mediante la cual las necesidades de un cliente son resueltas o reunidas con las capacidades de un proveedor. Una aplicación SOA está formada por un número de servicios interconectados cuyo objetivo es automatizar uno o varios procesos tanto de negocios (como una reserva de hotel, pagos con tarjeta a través de la web…) como tecnológicos que interactúan entre ellos, proporcionan la lógica necesaria para construir aplicaciones de una manera rápida.

Un prototipo de sistema STWS desarrollado en el NIST consiste en un cliente, un proveedor de los servicios (nodo de red inalámbrica) y nodos de sensores inalámbricos. El cliente y el proveedor de servicios se comunican entre sí a través de Simple Object Access Protocol (SOAP), utilizando mensajes del estándar IEEE 1451.0 y 1451.5-WiFi. El sistema prototipo fue probado con éxito a través de algunos estudios en determinados casos, y un estudio de caso de la lectura de datos del transductor se describe en detalle.

STWS se basa en la Arquitectura Orientada a Servicios (SOA). SOA es una evolución de la informática distribuida basada en la forma petición-respuesta de diseño para aplicaciones síncronas y asíncronas. El aspecto más importante de un servicio Web es la descripción del servicio mediante las funciones Web Services Description Language (WSDL) que describe los mensajes, los tipos y las operaciones de servicio Web y la forma en la que el servicio Web da las garantías para que se cumplan.

SOA está basada en estándares, es independiente de la plataforma, independiente del lenguaje y permite independencia en la interoperabilidad del

sistema operativo. Por lo tanto, la aplicación de la SOA a los sensores y redes de sensores es un paso importante hacia la configuración de los sensores como recursos reutilizables, permitiendo que sean detectables, accesibles y controlables a través de Internet. Los objetivos de STWS son permitir una Web residente en IEEE 1451 para hacer a los transductores inteligentes para ser detectables, accesibles y controlables mediante servicios Web a través de Internet y lograr así interoperabilidad basada en estándares para aplicaciones de transductores inteligentes.

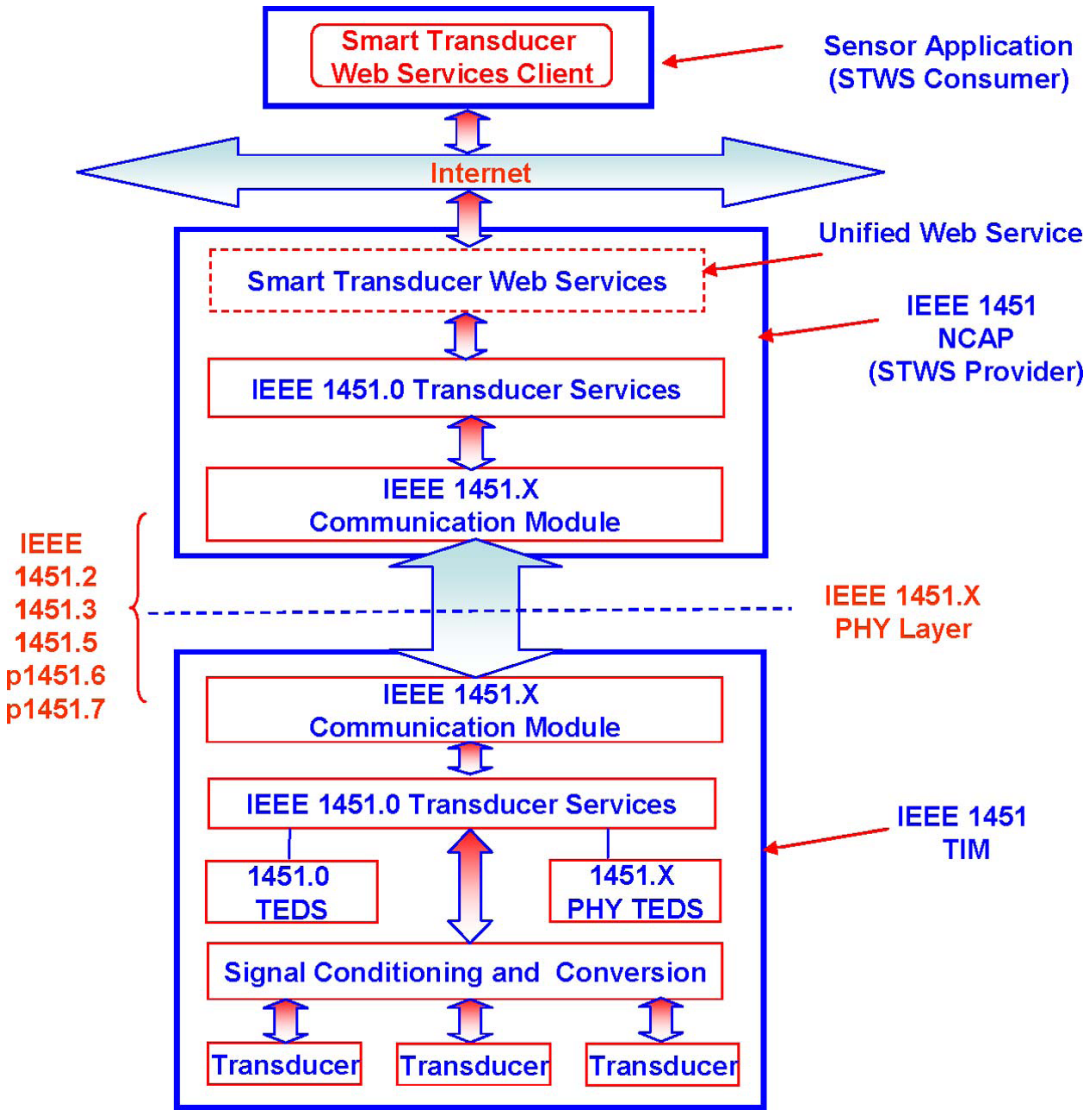


Ilustración 24: STWS en la arquitectura de IEEE 1451.

En la siguiente figura se muestra un ejemplo de STWS basado en interoperabilidad con WSDL. STWS, basado en el estándar IEEE 1451.0, ha sido definido usando funciones WSDL. El proveedor de servicios puede ser generado a partir del archivo WSDL utilizando herramientas de desarrollo de servicios Web en lenguajes de programación como Java. Por otra parte, las Web Service cliente también pueden ser generadas a partir del archivo WSDL permitiendo que se realice en diferentes lenguajes de programación como Java, C # o C + +. Los dispositivos consumidores de los servicios tales como sistemas de sensores de alerta, OGC-SWE o aplicaciones del sensor puede ser interoperables con el proveedor de servicios a través de mensajes SOAP / XML.

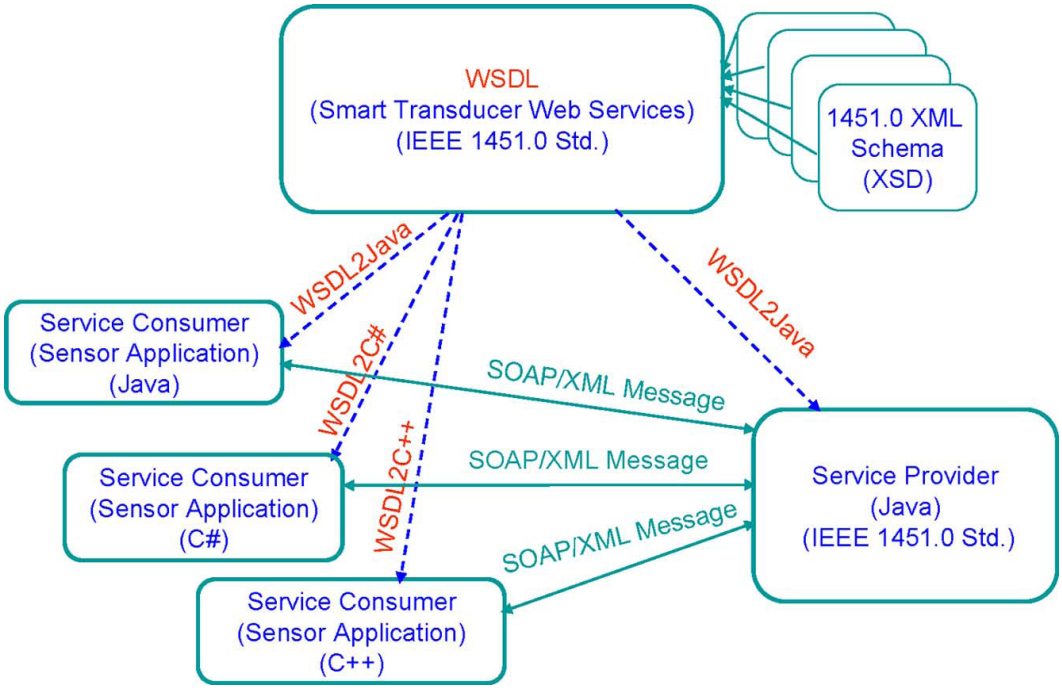


Ilustración 25: Interoperabilidad de STWS con las funciones WSDLL.

8.3.IEEE 1451.1

Se trata del primer miembro de la familia de IEEE 1451. Su objetivo era definir un modelo de objetos común para transductores inteligentes, junto con especificaciones de interfaz para los componentes del modelo, ya ha pasado mucho tiempo desde su creación y en la actualidad con la llegada de IEEE 1451.0 y STWS ha pasado a un segundo plano, perdiendo importancia y quedando en cierta parte "obsoleto", como se explicará a continuación.

El estándar IEEE 1451.1, en particular, define un modelo de objetos genéricos adecuados para representar cualquier transductor en red. Este modelo debe ser implementado en un procesador, llamado NCAP, que actúa como un puente entre los transductores y la red de comunicaciones. Por el lado del NCAP que se conecta con el transductor, una capa de abstracción proporciona funciones de alto nivel para comunicarse con los transductores. Este estándar recomienda la adopción de interfaces 1451.X pudiéndose adoptar incluso IEEE 1451.4. Por el lado de la red, una capa de abstracción proporciona servicios de alto nivel para manejar las peticiones de red. En el medio, la aplicación NCAP procesa los datos recibidos de ambas partes (la red transductor y la red usuario) y decide el siguiente estado del sistema. El modelo de objetos propuestos por el estándar 1451.1 está compuesto por una jerarquía de clases divididas en tres categorías principales:

- Bloque de clases destinadas a realizar el procesamiento de datos.
- Clases de componentes destinadas para encapsular los datos.
- Clases de servicios destinadas a controlar la comunicación entre NCAPs y realizar la sincronización de todo el sistema.

Los objetos creados a partir de estas clases son la base utilizada para generar la aplicación NCAP. La jerarquía es extensible mediante la adición de clases no pertenecientes a 1451.1 para satisfacer las necesidades particulares de una determinada aplicación. Los fabricantes de los NCAP procederán a elaborar el modelo de objetos para su producto e implementarlo como una biblioteca de

software reutilizable. El estándar 1451.1 ofrece dos modelos para la comunicación inter-NCAP: un modelo cliente/servidor en las comunicaciones one-to-one, y un modelo publish/subscribe para las comunicaciones one-to-many. Usando el modelo cliente / servidor, un cliente NCAP puede llamar una función en el servidor NCAP y quedarse cualquier resultado devuelto.

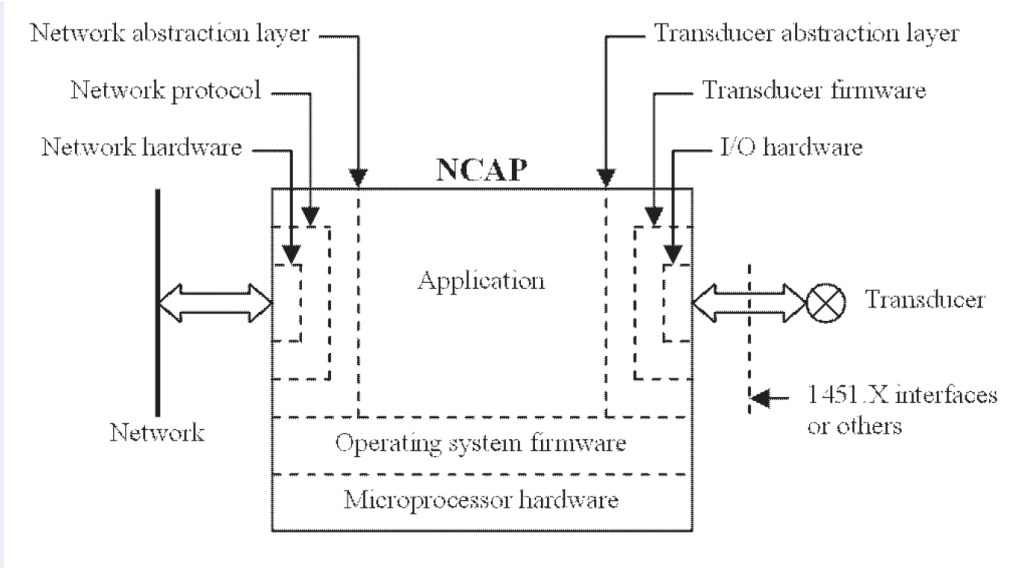


Ilustración 26: Arquitectura del IEEE 1451.1

Tres modos de llamada están disponibles:

- Modo síncrono: El cliente espera a que el servidor responda.
- Modo asíncrono: El cliente realiza la llamada remota y continúa su ejecución. Más tarde, se puede consultar si el servidor ha terminado la llamada, y si es así, solicitar los datos devueltos.
- Modo Send-and-Forget: Este modo es similar a la anterior. La única diferencia es que el cliente no se preocupa de los datos devueltos en absoluto.

Mediante el modelo publish/subscribe, el publisher NCAP puede enviar mensajes broadcast en la red siempre que tenga algún tipo de anuncio público que hacer. Estos mensajes, conocido como publicaciones, contienen metadatos que describen la sintaxis y la semántica de la información que transmiten. En base a estos metadatos, cualquier subscriber NCAP puede filtrar las publicaciones de su interés y quedarse sus datos adjuntos. Este modelo es una forma muy eficiente de organizar eventos distribuidos.

Es muy importante señalar que IEEE 1451.1 ha perdido peso dentro de la familia IEEE 1451, llegando a quedar, en cierta parte, obsoleto con la llegada en el año 2007 de IEEE 1451.0. Dicho 1451.0 define la arquitectura de las capas dentro del NCAP con las APIs que debe incluir y la interoperabilidad entre distintos IEEE 1451.X, los cuales tan sólo se recomiendan en 1451.1. El nivel de aplicación en 1451.0 se define a partir de HTTP (el cual se muestra más adelante) más sencillo que la forma 1451.1.

Por tanto, para una implementación hoy en día, es mejor opción utilizar como comunicación entre NCAP el HTTP de 1451.0 o el Smart Transducer Web Service.

8.4.IEEE 1451.0. Http

HTTP es un protocolo utilizado para transmitir información en la World Wide Web. HTTP es un protocolo cliente-servidor por el cual pueden comunicarse dos procesadores a través de una conexión TCP/IP. Un servidor HTTP es un programa que reside en un procesador quedando a la escucha de peticiones HTTP a través de un puerto. Un cliente HTTP abre una conexión TCP/IP con el servidor a través de un socket, transmite la petición y, a continuación, espera una respuesta del servidor. El modelo cliente-servidor es análogo a la comunicación "NCAP-user" que define esta norma. El NCAP se puede asimilar al "servidor", ya que sirve los datos por la red. El usuario puede ser comparado con el "cliente" ya que recibe datos de sensores desde el servidor y envía comandos y datos al servidor para accionar actuadores.

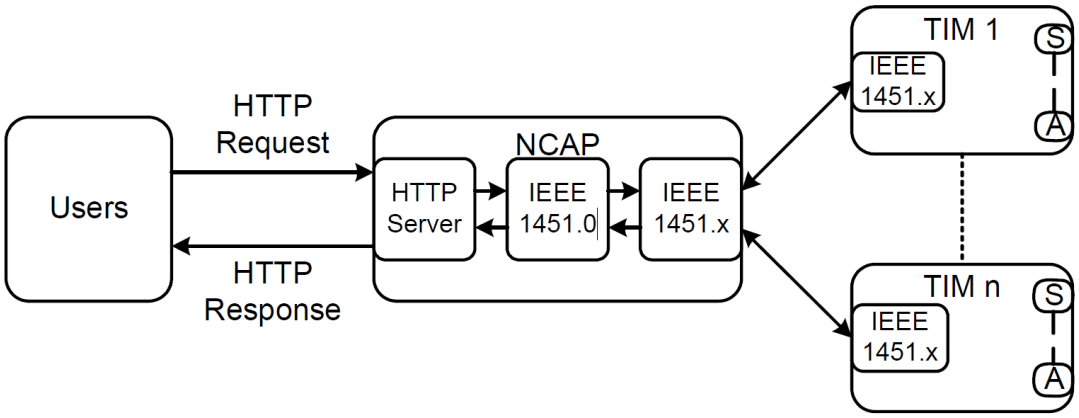


Ilustración 27: Arquitectura y forma de trabajo de IEEE 1451.0 para un nivel de aplicación implementado con HTTP API.

Solicitud HTTP desde el Cliente: El cliente HTTP envía un mensaje de solicitud de formato de acuerdo a las normas de HTTP estándar. En este mensaje se especifican los recursos desde los que el cliente desea recuperar información.

Respuesta HTTP del servidor: El servidor lee e interpreta la petición. Toma las acciones pertinentes y genera un mensaje de respuesta HTTP, que envía al cliente. El mensaje de respuesta indica si la solicitud se ha realizado correctamente y los datos que el cliente solicitó.

HTTP define ocho métodos de interacción con el servidor que son GET, POST, HEAD, PUT, DELETE, TRACE, y OPTIONS. El API definido en este estándar solamente utiliza los métodos GET y POST.

GET: Recupera la información que se identifica por la petición-URI. Se utiliza normalmente para recabar datos. La interacción puede ser una pregunta, una consulta, una operación de lectura o de búsqueda. En el estándar GET se puede utilizar para leer y escribir datos al transductor o para leer TEDS.

POST: Se utiliza para solicitar que el servidor acepte la entidad adjunta como datos para el recurso identificado por la petición-URI. La interacción se puede

ver como una orden, o una petición de permiso para realizar cambios del estado de los recursos (por ejemplo, una suscripción a un servicio). Este método puede utilizarse para enviar comandos o datos desde el usuario hacia el servidor o enviar y recibir datos al transductor o leer y escribir TEDS.

API HTTP IEEE 1451.0

El TSI es un API utilizado para permitir que aplicaciones de medición y control ejecutadas dentro de un NCAP accedan a la capa IEEE 1451.0. Este API contiene operaciones para leer y escribir TransducerChannels, leer y escribir TEDS, enviar comandos de configuración, control y operación al TIM. El TSI en la parte de la capa IEEE 1451.0 contiene cinco interfaces:

1. TIM Discovery
2. Transducer Access
3. Transducer Manager
4. TEDS Manager
5. Application Callback.

Los cuatro primeros son implementados por la capa IEEE 1451.0 y son llamados por las aplicaciones. Si la aplicación, en este caso el servidor HTTP, desea utilizar características avanzadas opcionales, deberá implementar el interfaz AppCallback que la capa IEEE 1451.0 llamará.

IMPORTANTE—Ningún miembro de IEEE 1451 define algún API AppCallback API.

El API HTTP de este estándar se focaliza principalmente en el acceso a los datos del transductor y TEDS utilizando el protocolo HTTP 1.1. En la figura 3 se muestra como se utiliza el protocolo HTTP para acceder a un NCAP que dispone de servidor HTTP.

1. Un usuario o cliente envía una petición HTTP hacia el servidor situado en el NCAP IEEE1451.0.

2. El Servidor HTTP recibe la petición, la procesa y llama a las funciones del APIIEEE 1451.0 correspondientes.

3. La capa IEEE 1451.0 utiliza el API IEEE1451.X para conseguir los resultados desde el TIM

4. El Servidor HTTP obtiene los resultados de las funciones del API IEEE 1451,0 y envía los resultados al usuario.

Formato del mensaje http

En esta sección se describirá el uso del protocolo HTTP para enviar mensajes desde un cliente remoto a un NCAP IEEE 1451.0 que actúa como servidor.

En la siguiente tabla se listan los formatos de un mensaje HTTP con ejemplos de posibles argumentos. Este mensaje se adhiere a la sintaxis del estándar HTTP URL (RFC 2616) como sigue:

http://<host>:<port>/<path>?<parameters>

Campo	Definición	Ejemplo
<host>:	La porción host de la sentencia incluirá el nombre de dominio del nodo IEEE 1451	<host> = "192.168.1.91"
<port>	El número de puerto es opcional siendo el 80 el puerto por defecto. Puede ser útil no utilizar el puerto por defecto, pero tiene como contrapartida que muchos routers, bloquean el acceso a puertos no estándar por seguridad.	<port>="80"
<path>	El path indica el camino hacia el comando.	<path>="1451/TransducerAccess/ReadData"
<parameters>	Indican los parámetros asociados al. comando.	<parameters>="timId=1&channelId=2&timeout=14&samplingMode=continuous&format=text"

Tabla 27: Descripción de los parámetros RESTFULL.

En la siguiente tabla se muestran los caminos permitidos en este estándar:

API	Nombre	PATH
Discovery API	TIMDiscovery	1451/Discovery/ TIMDiscovery
	TransducerDiscovery	1451/Discovery/ TransducerDiscovery
Transducer Acces API	ReadData	1451/TransduccerAccess/ReadData
	StartReadData	1451/TransduccerAccess/StartReadData
	MeasurementUpdate	1451/TransduccerAccess/MeasurementUpdate
	WriteData	1451/TransduccerAccess/WriteData
	StartWriteData	1451/TransduccerAccess/StartWriteData
TEDS Manager API	ReadTeds	1451/TEDSManager/ReadTeds
	ReadRawTeds	1451/TEDSManager/ReadRawTeds
	WriteTeds	1451/TEDSManager/WriteTeds
	WriteRawTeds	1451/TEDSManager/WriteRawTeds
	UpdateTedsCache	1451/TEDSManager/UpdateTedsCache
Transducer Manager API	SendCommand	1451/TransducerManager/SendCommand
	StartCommand	1451/TransducerManager/StartCommand
	CommandComplete	1451/TransducerManager/CommandComplete
	Trigger	1451/TransducerManager/Trigger
	StartTrigger	1451/TransducerManager/StartTrigger

Tabla 28: Interfaces que componen el HTTP API de IEEE 1451.0.

Formato de la respuesta HTTP

En la petición HTTP puede incluirse el formato en el que se desean recibir la respuesta en el argumento **responseFormat**. Los posibles formatos son XML, HTML y TEXT.

Formato de respuesta XML

Si se indica que el formato de respuesta sea XML, la respuesta del servidor http deberá estar formateada según el schema-XML asociado con el comando. El schema-XML completo puede obtenerse en:

http://grouper.ieee.org/groups/1451/0/1451HTTPAPI/

El nombre del fichero es SmartTransducerHTTPResponse.xsd.

Formato de respuesta HTML

Si se indica HTML como formato de respuesta, el servidor deberá formatear su salida como una página válida en formato HTTP 1.1. Este estándar no especifica el formateo ni la disposición de esta página, sin embargo todos los parámetros que se especifican en la respuesta deberán usar el formato de tag indicando para comando.

Formato de respuesta TEXT

En el formato TEXT cada valor individual devuelto será retornado separado del siguiente por la secuencia CARRIAGE RETURN/LINE FEED (CR/LF ASCII 13,10). Los valores serán retornados en el orden especificado por la semántica del comando.

Si se debe retornar un array, cada valor deberá ir separado por comas y el array deberá ser terminado por una secuencia CR/LF. Los arrays multidimensionales deberán ser retornados empezando por el ordinal de más a la derecha. Se utilizará una secuencia CR/LF cada vez que se complete un grupo de índices de más a la derecha. Este es el formato habitualmente llamado CSV.

Los enteros deberán ser retornados en formato <SIGNO><VALOR> donde signo es "+" o "-", y valor es un string que contiene uno o más caracteres entre "0" y "9".

Los números en coma flotante deberán ser retornados en notación científica de*<signo><cifras principales>.<mantisa>E<signo><exponente>.*

<Signo> es como se ha definido antes y <cifras principales>, <mantisa> y <exponente>es como se definió <valor>

Cualquier valor que represente una enumeración, deberá retornar su valor ordinal como un entero.

Cualquier valor que represente un string, deberá ir entre comillas. Unas comillas en el texto se deberán duplicar.

9. GUÍA DE IMPLEMENTACIÓN SOBRE IEEE 1451

9.1. INTRODUCCIÓN. CONSIDERACIONES INICIALES

Este apartado es una guía para realizar una implementación de IEEE 1451.

Lo que se quiere mostrar en este apartado no son soluciones directas para realizar una implementación concreta sino mostrar el camino lógico y los pasos que hay dar, así como el orden de las preguntas que uno se debe ir formulando en función de las condiciones iniciales de las cuales se parta o a las que se desee llegar.

Si se disponen de dispositivos hardware muy potentes o versátiles para llevar a cabo la implementación del estándar, puedes crear un código y funciones de programación de implementación las cuales serán mucho más fácil de reutilizar e incluso embeber en otro dispositivo hardware u otra aplicación realizando pequeños cambios tanto en la aplicación como en el código. Normalmente, esta situación no es la más común y la decisión de introducir un estándar de comunicaciones (en este caso de sensores inteligentes) suele tratarse de una necesidad a posteriori, es decir, debido a una ampliación en el sistema de comunicación o de obtener una nueva funcionalidad con el dispositivo hardware. Con esto quiero decir que normalmente, si las condiciones iniciales necesitadas en la aplicación no necesitan de un estándar para realizar su tarea, como es lógico no se implementa pero si posteriormente se desea que realice una comunicación con otro equipo o nuevas funciones para las cuales se necesita introducir un estándar (o al menos parte de él) es más complicado introducirlo.

Un ejemplo de esto, es lo que ocurrió en la historia de los televisores. Hacia la década de 1950, se creó el primer sistema de televisión a color, NTSC. Este sistema fue creado para que los nuevos televisores se pudieran ver a color y los antiguos (blanco y negro) pudieran seguir siendo utilizados. Para ello la anterior señal de televisión en blanco y negro fue modificada hasta conseguir obtener la

señal NTSC, permitiendo que los televisores antiguos no captaran la diferencia de señal, es decir, respetando la compatibilidad.

Queda claro por tanto que los pasos y cuestiones a decidir no son los mismos cuando se parte un dispositivo hardware dado, en el que se desee implementar el estándar, que cuando no importe el dispositivo hardware sobre el cual se vaya a realizar dicha implementación. Como tampoco es lo mismo realizar una implementación para un transductor con cable que cuando nos interese crear un entorno inalámbrico y habrá que elegir también el lenguaje de programación en cual se desee desarrollar la implementación teniendo en cuenta los recursos disponibles.

IEEE 1451.0 tiene como objetivo desarrollar un conjunto de funcionalidades comunes, los comandos y las TEDS, para la familia de estándares IEEE 1451 de transductores inteligentes. Esta funcionalidad será independiente de los medios de comunicación física, es decir, del Módulo de Comunicaciones empleado. Incluye las funciones básicas necesarias para controlar y gestionar los transductores inteligentes, protocolos de comunicación y los medios de comunicación independientes del Transductor Electronic Data Sheet (TEDS) formatos. Se define un conjunto de interfaces de programación de aplicaciones independientes (API) según sea necesario.

En este apartado no se va a explicar configuraciones y aspectos avanzados del IEEE 1451, como por ejemplo crear grupos de varios TIMs, utilización de proxy… sino que se va a centrar en los aspectos básicos y en lo mínimo que debe de cumplir para que el sistema cumpla la norma IEEE 1451.

Eje principal 1451.0

A día de hoy, el miembro más importante (y como se ha explicado y justificado anteriormente) del estándar IEEE 1451 es IEEE 1451.0, ya que es el encargado de recopilar y organizar al resto de los miembros y aunque anteriormente se podía implementar tan sólo el IEEE 1451.2 sin necesidad del

IEEE 1451.0, a día de hoy no tendría sentido puesto que este es el miembro que permite la mayor versatilidad al posibilitar que todos los miembros de la familia de IEEE 1451 sean compatibles entre ellos y además define perfectamente las funcionalidades del TIM y del NCAP. En el Stack de IEEE 1451 se observa perfectamente esta cuestión. Independientemente de la aplicación final que vaya a tener la red de sensores implementada, este miembro del estándar va estar siempre presente tanto en TIM como en el NCAP.

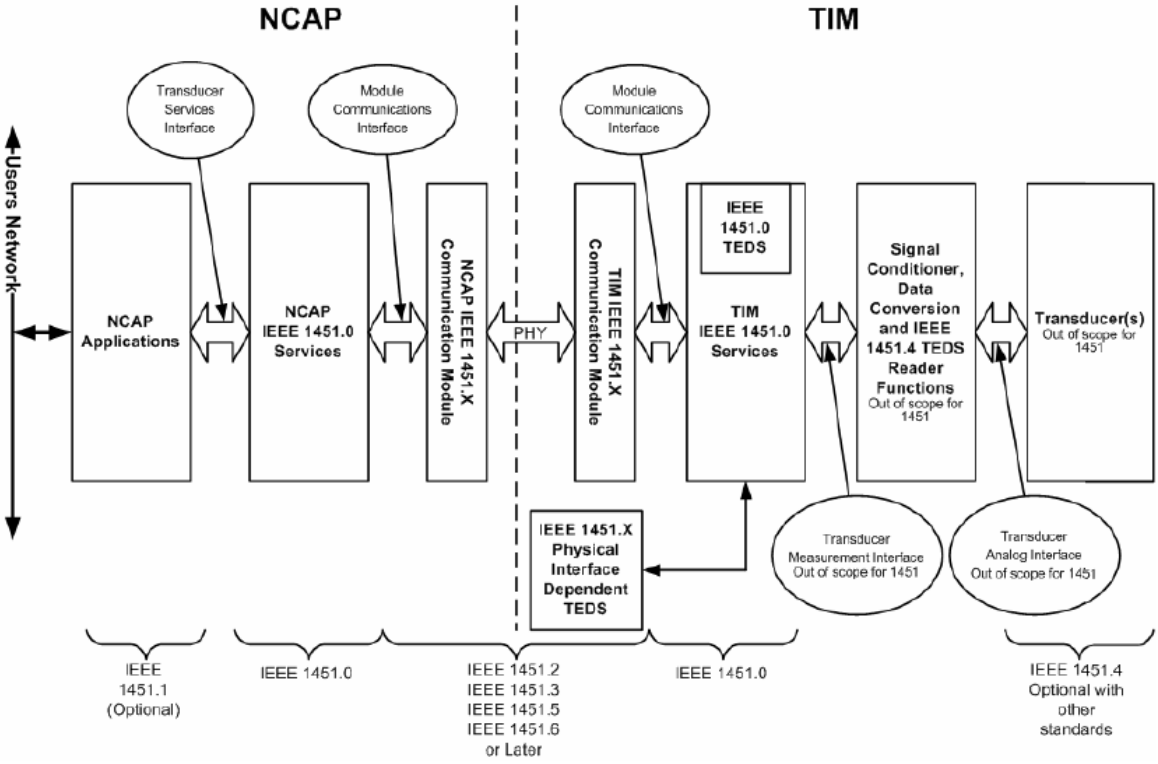


Ilustración 28: Capas en IEEE 1451.

El primer paso que se va a tomar es dividir la implementación del estándar en dos partes. Esta división va estar marcada por el dispositivo NCAP. Como se ha explicado anteriormente, este dispositivo es el "cerebro" de la red de sensores que desciende él, es decir, se encarga de gestionar tanto a los sensores inteligentes como a la información que transmiten dichos sensores a través de la red para posteriormente guardarla y evaluarla para tomar las decisiones que procedan. Al

fin y al cabo, el papel que juega el NCAP en IEEE 1451 es como el de un traductor, recibe los mensajes externos de la red de usuario que son enviados por un usuario, estos mensajes pueden ser para controlar el estado de un sensor o activar/desactivar un actuador. El NCAP analiza el mensaje y si no tiene la información necesaria en su memoria para responder al mensaje, lo "traduce" y lo envía al TIM para que obtenga la información. Una vez que dicho TIM obtiene la información (ya sea por lectura de la EEPROM, donde se encuentran las TEDS, o por consulta al sensor) le contesta al NCAP. Dicho NCAP vuelve a "traducir" el mensaje para poder mandárselo al usuario que se encuentra fuera de la red y el cual había realizado la petición de los datos.

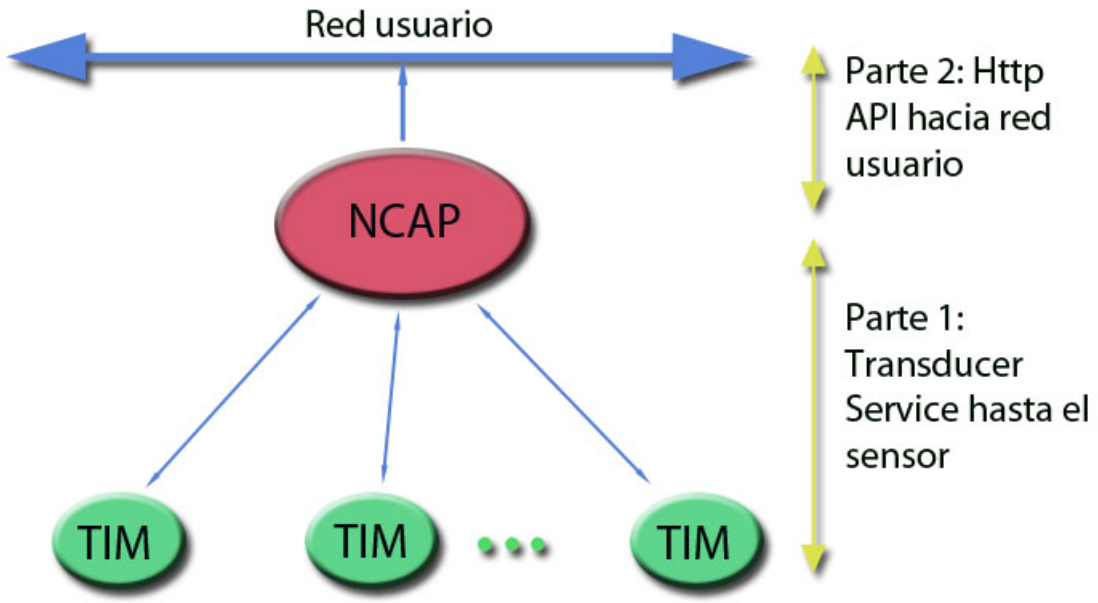


Ilustración 29: Se separa en dos grandes bloques el proceso de implementación

9.2.TEDS

Las TEDS son, como ya sabemos, las herramientas que almacenan y actualizan los datos de un sensor, TIM, transducer channel… Permiten hacer compatibles a los distintos sensores inteligentes que utilicemos en nuestra red. Son una herramienta muy poderosa que configuradas correctamente permiten una comunicación satisfactoria independientemente de la naturaleza de la magnitud que mida el sensor inteligente (es decir del tipo de sensor) y del fabricante del sensor.

Bajo mi punto de vista, se podría decir de una forma poco ortodoxa que las TEDS son, como ocurría en caso del NCAP explicado anteriormente, un "traductor" pero en este caso debemos de indicarle cómo va a ser el "lenguaje" entre el sensor y el estándar.

Tan sólo son suficientes cuatro TEDS para implementar el estándar.

9.2.1. Meta-TEDS

La función de una Meta-TEDS es hacer disponible al interfaz toda la información que necesita para tener acceso a cualquier TransducerChannel y la información común a todos los TransducerChannels. Esta TEDS tiene que estar alojada en el TIM al que identifique, es decir, la Meta-TEDS identifica e informa sobre un determinado TIM, no de un transductor. Debe de haber una de estas TEDS por cada TIM que haya en la red.

Los campos fundamentales son un identificador "UUID", que sirve para identificar de forma única a dicho TIM, también contiene parámetros de "worst timing parameters" y *"time out values"* que van a servir para identificar cuando el TIM no está respondiendo.

La información que contiene esta TEDS es sobre identificación y parámetros de funcionamiento de un determinado TIM, por lo que no habrá necesidad de cambiar la información (salvo excepción) de dicha TEDS. Por tanto, a la hora de implementar la Meta-TEDS se realizará de tal forma que se aloje en memoria FLASH (ROM), es decir, implementarla como constantes de forma que se utilice la menor memoria RAM posible, ya que esta suele ser más limitante en microprocesadores de bajo consumo como el msp430..

A continuación se muestra la tabla que especifica los atributos y parámetro que debe tener una Meta-TEDS así como el nombre de los parámetros así como su tamaño.

Field type	Field name	Description	Data type	# octets
—		Length	UInt32	4
0–2	—	Reserved	—	—
3	TEDSID	TEDS Identification Header	UInt8	4
4	UUID	Globally Unique Identifier	UUID	10
5–9	—	Reserved	—	—
Timing-related information				
10	OholdOff	Operational time-out	Float32	4
11	SHoldOff	Slow-access time-out	Float32	4
12	TestTime	Self-Test Time	Float32	4
Number of implemented TransducerChannels				
13	MaxChan	Number of implemented TransducerChannels	UInt16	2
14	CGroup	**ControlGroup information sub-block**	—	—
Types 20, and 21 define one ControlGroup.				
20	GrpType	ControlGroup type	UInt8	1
21	MemList	ControlGroup member list	array of UInt16	NTc
15	VGroup	**VectorGroup information sub-block**	—	—
Types 20 and 21 define one VectorGroup.				
20	GrpType	VectorGroup type	UInt8	1
21	MemList	VectorGroup member list	array of UInt16	NTv
16	GeoLoc	**Specialized VectorGroup for geographic location**	—	—
Types 24, 20, and 21 define one set of geographic location information.				
24	LocEnum	An enumeration defining how location information is provided	UInt8	1
20	GrpType	VectorGroup type	UInt8	1
21	MemList	VectorGroup member list	array of UInt16	NTv
17	Proxies	**TransducerChannel proxy definition sub-block**	—	—
Types 22, 23, and 21 define one TransducerChannel proxy.				
22	ChanNum	TransducerChannel number of the TransducerChannel proxy	UInt16	1
23	Organiz	TransducerChannel proxy data-set organization	UInt8	1
21	MemList	TransducerChannel proxy member list	array of UInt16	NTp
18–19	—	Reserved	—	—
25–127	—	Reserved	—	—
128–255	—	Open to manufacturers	—	—
—		Checksum	UInt16	2

Tabla 29: Todos los campos que componen el Data Block de la META-TEDS.

No obstante, para realizar una implementación sencilla sería suficiente con implementar los siguientes campos que se muestran en la siguiente tabla:

Campo	Nombre	Descripción	Tipo dato	#Octetos
-		Longitud Data block	UInt32	4
3	TEDSID	Identficación cabecera TEDS	UInt8	4
4	UUID	Identificador único TIM	UUID	10
10	OholdOff	Time-out de una operación	Float32	4
11	SHoldOff	Tiempo espera entre operaciones	Float32	4
12	TestTime	Tiempo para ejecutar self-test	Float32	4
13	MaxChan	Número de TransducerChannels	UInt16	2
-		CheckSum del Data block	Uint32	2

Tabla 30: Campos mínimos que se deben implementar en una META-TEDS.

Como se observa en la tabla, se ha añadido al inicio y al final la longitud del "data block" y el checksum del "data block" respectivamente. Recordar, como se ha explicado anteriormente en el apartado de las TEDS, que cada campo de la tabla establece el tipo (Type) de la TLV (Type Length Value) y hay que añadir el Length que es la longitud que tiene el parámetro y el Value el valor (Más información en el apartado TEDS).

El UUID es un tipo de datos que consta de 32 bits con números enteros (Int16), considerado como una entidad global.

UUID		
Campo	**Bits**	**Descripción**
Norte/Sur	1	Si se encuentra en Norte (1) o Sur (0)
Latitud	20	Latitud en la que se encuentra
Este/Oeste	1	Este (1) u Oeste (0)
Longitud	20	Longitud
Fabricante	4	Información del fabricante
Año	12	Año de fabricación
Sequencia/Fecha	22	Fecha de fabricación junto con el nº serie

Tabla 31: Campos y descripción de UUID.

9.2.2. TransducerChannel-TEDS

Se trata, bajo mi punto de vista y tras haber estudiado y comprendido las Transducer Electronic Data Sheets, de la TEDS más importante. Para realizar una buena implementación es esencial que esa TEDS se encuentre bien definida. Como recordamos esta TEDS es la que controla el transductor en la comunicación TIM-Transducer (denominando esta comunicación TransducerChannel, porque va desde el TIM hasta el transductor).Esta TEDS también es la responsable de especificar si hace falta implementar alguna otra TEDS opcional.

Da los parámetros físicos que se están midiendo o controlando, el rango sobre el cual el TransducerChannel opera, las características de la E/S digitales, el modo operativo de la unidad y la información de los tiempos. Por ejemplo, un campo que hay que especificar siempre y que es muy importante es el de Calibración (si hace falta una calibración externa a la TEDS, o incluye una Calibration TEDS, o si el propio módulo realiza la calibración…)

A continuación se muestra una tabla en la podemos observar todos los campos que contiene esta TEDS.

Field	Field name	Description	Type	# octets
—		TEDS length	UInt32	4
0–2	—	Reserved	—	—
3	TEDSID	TEDS Identification	UInt8	4
4-9	—	Reserved	—	—
TransducerChannel related information				
10	CalKey	Calibration key	UInt8	1
11	ChanType	TransducerChannel type key	UInt8	1
12	PhyUnits	Physical Units	UNITS	11
50	UnitType	Physical Units interpretation enumeration	UInt8	1
51	Radians	The exponent for Radians	UInt8	1
52	SterRad	The exponent for Steradians	UInt8	1
53	Meters	The exponent for Meters	UInt8	1
54	Kilogram	The exponent for Kilograms	UInt8	1
55	Seconds	The exponent for Seconds	UInt8	1
56	Amperes	The exponent for Amperes	UInt8	1
57	Kelvins	The exponent for Kelvins	UInt8	1
58	Moles	The exponent for Moles	UInt8	1
59	Candelas	The exponent for Candelas	UInt8	1
60	UnitsExt	TEDS access code for units extension	UInt8	1
13	LowLimit	Design operational lower range limit	Float32	4
14	HiLimit	Design operational upper range limit	Float32	4
15	OError	Worst-case uncertainty	Float32	4
16	SelfTest	Self-test key	UInt8	1
17	MRange	Multi-range capability	UInt8	1
—		Data converter-related information	—	—
18	Sample		—	—
40	DatModel	Data model	UInt8	1
41	ModLenth	Data model length	UInt8	1
42	SigBits	Model significant bits	UInt16	2
19	DataSet			
43	Repeats	Maximum data repetitions	UInt16	2
44	SOrigin	Series origin	Float32	4
45	StepSize	Series increment	Float32	4
46	SUnits	Series units	UNITS	11
47	PreTrigg	Maximum pre-trigger samples	UInt16	2
Timing-related information				
20	UpdateT	TransducerChannel update time (t_u)	Float32	4
21	WSetupT	TransducerChannel write setup time (t_{ws})	Float32	4
22	RSetupT	TransducerChannel read setup time (t_{rs})	Float32	4
23	SPeriod	TransducerChannel sampling period (t_{sp})	Float32	4
24	WarmUpT	TransducerChannel warm-up time	Float32	4
25	RDelayT	TransducerChannel read delay time (t_{ch})	Float32	4
26	TestTime	TransducerChannel self-test time requirement	Float32	4
Time of the sample information				
27	TimeSrc	Source for the time of sample	UInt8	1
28	InPropDl	Incoming propagation delay through the data transport logic	Float32	4
29	OutPropD	Outgoing propagation delay through the data transport logic	Float32	4
30	TSError	Trigger-to-sample delay uncertainty	Float32	4
Attributes				
31	Sampling	Sampling attribute	—	—
48	SampMode	Sampling mode capability	UInt8	1
49	SDefault	Default sampling mode	UInt8	1
32	DataXmit	Data transmission attribute	UInt8	1
33	Buffered	Buffered attribute	UInt8	1
34	EndOfSet	End-of-data-set operation attribute	UInt8	1
35	EdgeRpt	Edge-to-report attribute	UInt8	1
36	ActHalt	Actuator-halt attribute	UInt8	1

Field	Field name	Description	Type	# octets
Sensitivity				
37	Directon	Sensitivity direction	Float32	4
38	DAngles	Direction angles	Two Float32	8
Options				
39	ESOption	Event sensor options	UInt8	1
61–127	—	Reserved	—	—
128–255	—	Open to manufacturers	—	—
—		Checksum	UInt16	2

Tabla 32:Todos los campos que componen el Data Block de la Transducer Channel TEDS

Como se observa se trata de una tabla bastante compleja y larga, no obstante, no hace falta implementar todos los campos, tan sólo los que sean necesarios para tu transductor ya que según sea sensor, actuador o detector de eventos (event-sensor) y según sea la magnitud con la que trabaje (humedad, temperatura…) se rellenarán unos campos u otros.

A continuación, después de haber comprendido el estándar, se ha realizado la siguiente tabla, la cual es un resumen de la anterior y sólo contiene los campos necesarios para una situación típica, como por realizado una búsqueda de ejemplos de TEDS en distintas implementaciones y ejemplo, un sensor de temperatura:

Campo	Nombre	Descripción	Tipo Dato	octetos
-		Longitud Data block	Uint32	4
3	TEDSID	TransducerChannel TEDS Identification	UInt8	1
10	CalKey	Calibration Key	UInt8	1
11	ChaType	TRansducerCahnnel Type Key	UInt8	1
12	PhyUnits	Physical Units	UNITS	11
13	LowLimit	Design operational lower limit	Float32	4
14	HiLimit	Design operational upper limit	Float32	4
15	OError	Uncertainty under worst-case conditions	Float32	4
16	SelfTest	Self-test Key	UInt8	1
18	Sample		-	-
40	DatModel	Data Model	UInt8	1
41	ModLenth	Data Model Length	UInt8	1
42	SigBits	Model significant bits	UInt16	2
20	UpdateT	TransducerChannel update time	Float32	4
22	WSetupT	TransducerChannel read setup time	Float32	4
23	SPeriod	TransducerChannel sampling period	Float32	4
24	WarmUpT	TransducerChannel warm-up time	Float32	4
25	RDelayT	TransducerChannel read delay time	Float32	4
26	TestTime	TransducerChannel self-test time requirement	Float32	4
31	Sampling	Sampling attribute	-	-
48	SampMode	Sampling mode capability	Uint8	1
-		Checksum	Uint16	2

Tabla 33: Campos mínimos que se van a implementar en una Transducer Channel TEDS.

9.2.3. User's transducer name TEDS

Esta TEDS es obligatoria para los TIMs y es recomendable para todos los transductores. El User´s TransducerName TEDS proporciona un lugar para almacenar el nombre por el cual el sistema o el usuario final conocerán a este transductor. Es importante comprender que el objetivo de esta TEDS es soportar los "Object Tags" de IEEE 1451.1. Como en las TEDS anteriores, se muestra la tabla que especifica el estándar para su implementación con todos sus campos posibles.

Field type	Field name	Description	Data type	# octets
—		Length	UInt32	4
0–2	—	Reserved	—	—
3	TEDSID	TEDS Identification Header	UInt8	4
4	Format	Format description of this TEDS	UInt8	1
5	TCName	TIM or TransducerChannel Name	—	
—		Checksum	UInt16	2

Tabla 34: Campos que componen el data Block de la User Transducer Name TEDS.

Como se puede observar en esta TEDS, al margen del campo de identificación de cabecera de la TEDS (el cual es obligatorio en todas las TEDS como sabemos), tan sólo tiene dos campos más. El campo "Format" y el "TCName".

En este caso, cuando se realice la implementación de esta TEDS es necesario implementar los tres campos tal y como establece la norma. El "data block" de esta TEDS ocupa 11 bytes.

9.2.4. PHY TEDS

Está información que contiene esta TEDS es esencial, puesto que es la que dice (entre otras cosas) cómo va a ser la comunicación entre los dispositivos del sistema, NCAP y TIMs.

Contiene información sobre el medio físico por el cual se va a realizar la comunicación entre un TIM y un NCAP, dentro del Módulo de Comunicaciones. La función es hacer disponible al interfaz toda la información que necesita para tener acceso a cualquier canal, más la información común a todos los canales. Como sabemos, esta TEDS no la especifica IEEE 1451.0 si no que es responsabilidad de IEEE 1451.X (pudiendo ser "X" 2, 3, 5, 6 o 7). En nuestro caso nos hemos centrado en IEEE 1451.5, la familia que se ocupa de los protocolos y comunicaciones inalámbricas.

La siguiente tabla muestra todos los campos que contiene una PHY TEDS.

Field type	Field name	Description	Type	# octets	Contents
—		TEDS length	UInt32	4	
3	TEDSID	TEDS Identification Header	UInt8	4	
4–9		Reserved			
10	Radio	Radio Type	UInt8	1	See Table 4
11	MaxBPS	Max data throughput	UInt32	4	Bits per second
12	MaxCDev	Max Connected Devices	UInt16	2	Maximum number of devices that may be simultaneously operational with this device
13	MaxRDev	Max Registered Devices	UInt16	2	Maximum number of devices that may be simultaneously registered with this device
14	Encrypt	Encryption	UInt16	2	Encryption type and key length
15	Authent	Authentication	Boolean	1	TRUE = Authentication supported FALSE = not supported
16	MinKeyL	Min Key Length	UInt16	2	Minimum key length for security functions
17	MaxKeyL	Max Key Length	UInt16	2	Maximum key length for security functions
18	MaxSDU	Max SDU Size	UInt16	2	Maximum payload size for transfer between devices
19	MinALat	Min Access Latency	UInt32	4	Minimum period for initiating a first transmission to an unconnected device
20	MinTLat	Min Transmit Latency	UInt32	4	The period for transmitting an N-octet payload, where N is the minimum payload size
21	MaxXact	Max Simultaneous Transactions	UInt8	1	Maximum number of queued, uncompleted commands
22	Battery	Device is battery powered	UInt 8	1	A nonzero value indicates battery power
23	RadioVer	Radio Version #	UInt16	2	A zero value indicates unknown
24	MaxRetry	Maximum Retries before Disconnect	Uint16	2	A zero value indicates never disconnect
25–31		Reserved			
32–41		Bluetooth Protocol Specific			See Table 7
42–47		Reserved			
48–54		802-11 Radio Specific			
55–63		Reserved			
64–80		ZigBee Protocol Specific			
81–95		Reserved			
96–103		6LoWPAN Protocol Specific			
104–127		Reserved			
128–255		Open to manufacturers			
—		Checksum	UInt16	2	

Tabla 35: Campos que componen el Data Block de una Phy TEDS.

Esta tabla puede resumirse eliminando ciertos campos, que pueden no ser necesarios en todas las situaciones de implementación.

Campo	Nombre	Descripción	Tipo Dato	# octetos
-		Longitud Data block	Uint32	4
3	TEDSID	TransducerChannel TEDS Identification	UInt8	1
10	Radio	Radio Type	Uint8	1
11	MaxBPS	Max data throughput	Uint32	4
12	MaxCDev	Max Connected Devices	Uint16	2
13	MaxRDev	Max Registered Devices	Uint16	2
14	Encrypt	Encryption	Uint16	2
15	Authent	Authentication	Boolean	1
18	MaxSDU	Max SDU Size	Uint16	2
19	MinAlat	Min Access Latency	Uint32	4
20	MinTLat	Min Transmit Latency	Uint32	4
21	MaxXact	Max Simultaneous Transactions	Uint8	1
22	Battery	Device is battery powered	Uint8	1
23	RadioVer	Radio Version #	Uint16	2
24	MaxRetry	Maximum Retries before Disconnect	Uint16	2
-		Checksum	Uint16	2

Tabla 36: Campos mínimos que tendremos que implementar en una Phy TEDS de IEEE 1451.5.

También es cierto, que la gran parte de las TEDS de este tipo hacen uso de los campos que deja el estándar para el fabricante o para la tecnología que utilice como comunicación. Por ejemplo, en el caso de usar Zigbee se añaden campos para

controlar parámetros como la frecuencia de uso, la modulación empleada, la antena que lleva, la batería que utiliza, rango máximo, los canales usados…

Con todo esto el tamaño del "data block" de estas TEDS está comprendido entre 38 bytes como mínimo pero típicamente en torno a 54 bytes.

9.2.5. TEDS Opcionales

Por otra parte, aunque no son obligatorias, muchas implementaciones de IEEE 1451 hacen uso de otras TEDS. Sobre todo se tratan de TEDS que realizan alguna corrección de datos. A continuación, se muestran las más comunes y usadas.

Calibration TEDS

Calibration TEDS contiene información acerca de los parámetros de calibración, y el intervalo de calibración de un transductor. Asimismo, sienta las constantes necesarias para convertir los datos del transductor en una forma de unidades de ingeniería para los sensores o para convertir datos en unidades de ingeniería a la forma requerida por un actuador.

A continuación se muestra la tabla que proporciona el estándar con los campos que puede contener.

Field type	Field name	Description	Type	Required /optional	Data length in octets
—		TEDS length	UInt32		4
0–2	—	Reserved	—	—	—
3	TEDSID	TEDS identifier	UInt8	R	4
4–9	—	Reserved	—	—	—
—		Calibration date related information		—	—
10	LstCalDt	Last calibration date-time	TimeInstance	O	8
11	CalInrvl	Calibration interval	TimeDuration	O	8
Units information					
12	SIConvrt	SI units conversion constants	—	O	—
30	SISlope	SI units conversion slope	Float32	O	4
31	Intrcpt	SI units conversion intercept	Float32	O	4
Operational limits and uncertainty					
13	LowLimit	Operational lower range limit	Float32	O	4
14	HiLimit	Operational upper range limit	Float32	O	4
15	OError	Operational uncertainty	Float32	O	4
Mathematical conversion to be performed on the data before or after correction					
16	OConvert	Post-conversion operation	UInt8	O	1
17	IConvert	Pre-conversion operation	UInt8	O	1
TLV tuple 20 is used when the linear method of conversion is used.					
20	LinOnly	This field is used when only a linear single section conversion is required	—	O	—
TLV tuple 21 provides a description of one TransducerChannel that is involved in the correction specified in this TEDS					
21	XdcrBlk	TransducerChannel description	—	O	—
40	Element	Element number	UInt16	O	1
41	ChanNum	Correction input TransducerChannel	UInt16	O	1
42	ChanKey	Correction input TransducerChannel key	UInt8	O	1
43	Degree	TransducerChannel degree	UInt8	O	1
44	STable	Segment boundary values table	—	—	—
45	OTable	Segment offset values table	Float32Array	O	Note 1
46	LoBndry	Array of low boundary limits	Float32Array	O	Note 1
47	HiBndry	High boundary limit	Float32	O	4
TLV tuple 22 provides a set of coefficients for one segment of the correction specified in this TEDS					
22	CoefBlk	Multinomial coefficient	—	O	—
50	CellNum	Cell number of the segment to which this set of coefficients apply	UInt8	O	1
51	CoefSet	The set of coefficients that applies to this cell	Float32Array	O	Note 2
18–19	—	Reserved	—	—	—
23–29	—	Reserved	—	—	—
48–49	—	Reserved	—	—	—
52–127	—	Reserved	—	—	—
128–255	—	Open to manufacturers	—	—	—
—		Checksum	UInt16	—	2

Tabla 37: Campos que componen el Data Block de una Calibration TEDS.

El único campo obligatorio es el "3" (Identificador de TEDS), el resto es opcional según las necesidades que se tenga para realizar la implementación. La situación ideal sería que esta TEDS fuera proporcionada por el fabricante del sensor ya que él es el que sabe de antemano cómo realiza la medida su sensor. Si a cada sensor se le adjunta una Calibration TEDS, la cual convierta las medidas del sensor a las del sistema SI, se daría un gran paso para la estandarización de los sensores (actuadores y demás…) para IEEE 1451. En algunas ocasiones, dependiendo del tipo y naturaleza del sensor, se ofrecen un "CD-ROM" con ciertos drivers o con una interfaz gráfica, sería positivo que se adjuntara esta TEDS para tener más facilidad en usar dicho sensor como un dispositivo de la red de sensores inteligentes de IEEE 1451. Es cierto, que se trata de una TEDS opcional, puesto que la TransducerChannel TEDS permite hacer una pequeña "calibración" pero esta TEDS tiene también campos para configurar el papel y las condiciones del transductor en la red.

Frequency Response TEDS

La función de FrequencyResponse TEDS es hacer disponible la información de la frecuencia del TransducerChannel al usuario. FrequencyResponse TEDS proporciona una caracterización de la respuesta en frecuencia y la en fase del TransducerChannel. Especialmente tiene importancia en el muestreo de la señal al pasar de la señal analógica a digital, que puede derivar en problemas de aliasing. Se trata sobre todo de una TEDS informativa que se lleva a cabo en varias implementaciones teniendo sentido cuando se tenga pensado hacer algún tipo de tratamiento o acondicionamiento de la señal del sensor.

Field type	Field name	Description	Type	# octets
—		TEDS length	UInt32	4
0–2	—	Reserved	—	—
3	TEDSID	TEDS identifier	UInt8	4
4–9	—	Reserved	—	—
10	RefFreq	Reference frequency	Float32	4
11	RefAmp	Test amplitude	Float32	4
12	RefPhase	Phase at the reference frequency	Float32	4
The following fields comprise a structure that defines one data point. The structure is repeated n times, once for each data point				
13	Points	Points in the table	—	—
14–127	—	Reserved	—	—
128–255	—	Open to manufacturers	—	—
—	—	Checksum	UInt16	2

Tabla 38: Campos que componen el Data Block de una Frecuency Response TEDS.

Transfer Function TEDS

Esta TEDS opcional está muy ligada con la anterior. Proporciona una serie de constantes que pueden ser usadas para describir la función de transferencia del transductor. Los factores que afectan a la función de transferencia son el sensor, el acondicionamiento de señal analógica, el filtro de anti-aliasing y el tratamiento de la señal digital.

Field	Field name	Description	Type	# octets
—		TEDS length	UInt32	4
0–2	—	Reserved	—	—
3	TEDSID	TEDS identification header	UInt8	4
4–9	—	Reserved	—	—
Transfer function related information				
10	RefFreq	Reference frequency	Float32	4
11	OneZero	Single zero TF_SZ	Float32	4
12	OnePole	Single pole TF_SP	Float32	4
13	ZeroPole	Single zero with a dependant pole	—	—
14	PoleZero	Single pole with a dependent zero	—	—
15	ComplexZ	Complex zero	—	—
16	ComplexP	Complex pole	—	—
17	OneZZPol	Single zero at zero and a single pole	—	—
18	CRSlope	Constant Relative Slope	Float32	4
19	SampleT	Sample/Delay Time	Float32	4
20	DependP	Single pole dependent on a zero TF_SPm (x)	Float32	4
21	DependZ	Single zero dependent on a pole TF_SZm (x)	Float32	4
22	ComplexZF	Complex zero frequency	Float32	4
23	ComplexZQ	Complex zero quality factor	Float32	4
24	ComplexPF	Complex pole frequency	Float32	4
25	ComplexPQ	Complex pole quality factor	Float32	4
26–29	—	Reserved	—	—
30	NCoeff	Numerator coefficients (A0, A1, … An)	Array of Float32	n*4
31	DCoeff	Denominator coefficients (B0, B1, … Bm)	Array of Float32	m*4
32–127	—	Reserved	—	—
12–255	—	Open for manufacturers use	—	—
		Checksum		
NOTE—The number of coefficients, n or m, for fields 30 and 31 may be determined by dividing the tuple length by 4.				

Tabla 39: Campos que componen el Data block de una Transfer Function TEDS.

9.3. COMANDOS

Como se ha explicado, los comandos son la parte del mensaje en la que se envía qué es lo queremos obtener en la comunicación o qué significa la información que se envía, a fin de cuentas es un identificador que va ejecutar una serie de funciones y métodos contenidas en las interfaces de una API.

Existen bastantes comandos en el estándar y para realizar una implementación sencilla no tiene sentido implementar la totalidad de los mismos.

De acuerdo al estándar, el mínimo de los comandos que deben implementarse para que el sistema sea reconocido como compatible con IEEE 1451

se muestra a continuación, enunciando las clases de los comandos (class commads, en negrita) y sus respectivos *function commands* por los que se encuentran compuestos:

CommonCommands

- Query TEDS
- Read TEDS segment
- Write TEDS segment
- Update TEDS
- Run self-test
- Write service request mask
- Read service request mask
- Read status-event register
- Read status-condition register
- Clear status-event register
- Write status-event protocol state
- Read status-event protocol state

Transducer idle state commands

- AddressGroup definition

Transducer operating state commands

- Trigger command

Transducer idle or operating state commands

- TransducerChannel Operate
- TransducerChannel Idle
- Read TransducerChannel trigger state
- Read AddressGroup assignment

TIM sleep state commands

- Wake-up

TIM active state commands

- Read TIM version
- TIM Sleep (opcional pero es muy aconsejado)
- Store operational setup
- Recall operational setup
- Read IEEE 1451.0 version

TIM any state commands

- Reset

No obstante, en los escasos documentos y reportes encontrados en los que se ha llevado a cabo una implementación del estándar IEEE 1451 para tecnologías inalámbricas (Zigbee, WiFi, Bluetooth y 6loWPAN) y además se ha explicado en dicho documento los comandos llevados a cabo en la implementación, se ha observado como son menos los implementados que los necesarios según el estándar (a continuación, comandos implementados en el trabajo **"A ZigBee wireless sensor network compliant with the IEEE1451 standard"** de *Jorge Higuera, Jose Polo, Manel Gasulla, Instrumentation, Sensors and Interfaces Group, Universitat Politécnica de Catalunya*)

CommonCommands

- Query TEDS
- Read TEDS segment
- Write TEDS segment
- Update TEDS
- Run self-test

Transducer operating state commands

- Read TransducerChannel
- Write TransducerChannel

TIM active state commands

- Read TIM version
- Store operational setup
- Read IEEE 1451.0 version

Dada la escasez de implementaciones llevadas a cabo para tecnologías inalámbricas en dispositivos *hardware* de bajo consumo y en situaciones en las que se busque una implementación sencilla, es más conveniente llevar a cabo esta segunda lista de comandos, ya que mantiene las funcionalidades básicas de la red (como la gestión y control de las TEDS, y el control de los Transducer Channels).

A continuación, se van definir las funciones y métodos que se deben de implementar en la API del Módulo de Comunicaciones y la API Transducer Service para un caso estándar de implementación. Es muy importante darse cuenta de que estas APIs se usan para llevar a cabo (en la mayoría de las ocasiones) las tareas y peticiones de los comandos.

9.4. MODULO COMUNICACIONES

IEEE 1451.0 caracteriza cómo debe de ser el Módulo de Comunicaciones de forma general, es decir, explica qué interfaces contiene y cuáles son las funciones y métodos que dichas interfaces contienen así como el nombre y tipo de dato de cada una pero es responsabilidad del respectivo IEEE 1451.X definir cómo van a ser y qué van a hacer exactamente cada función puesto que puede variar de un miembro a otro o de una tecnología a otra (el módulo de comunicaciones de Zigbee cambia respecto al de Bluetooth aunque ambos pertenecen al miembro IEEE 1451.5).

Metafóricamente, se podría decir que el Módulo de Comunicaciones está formado por el sistema de carreteras, peajes y aduanas por las cuales va a viajar la información que se desea transmitir o recibir.

Ahora bien, como se ha explicado anteriormente, existen cuatro aspectos a tener en cuenta de cómo puede ser dicha comunicación, ahora se hará una valoración de cada uno para realizar una implementación:

- **One-Way** o **Two-Way**
    - *One-Way* se produce cuando el transmisor envía un mensaje al receptor terminando la comunicación. *Two-Way*, por su parte, hace que el transmisor envíe un mensaje al receptor y éste a su vez le contesta.

  A la hora de implementar nuestra red es crucial saber cómo queremos que se produzca la comunicación entre nuestros dispositivos (normalmente NCAP-TIM y TIM-NCAP). Según la finalidad de nuestra red decidiremos realizar una comunicación ONE-WAY o si merece más la pena realizar una TWO-WAY. Si por ejemplo, el cometido de nuestra red es controlar si se producen incendios en los bosques será suficiente con implementar sensores de eventos (event-sensors, que sean detectores de humo) que cuando detecten valores de CO_2 fuera de un valor predeterminado informen al NCAP. En este caso la comunicación es suficiente llevarla a cabo mediante ONE-WAY. Esa medida "anómala" quedará reflejada en la respectiva TEDS que se encuentra alojada en la memoria EEPROM (típicamente) y posteriormente enviada a la caché de las TEDS que se encuentra en el NCAP. A este tipo de red le llamaremos "red pasiva" puesto que no requiere un control o gestión desde un usuario remoto.

  Si en cambio, se tiene previsto que nuestra red reciba mensajes y peticiones provenientes de la red usuario, teniendo una mayor interactividad con el sensor o actuador, sería más conveniente tener una comunicación TWO-

WAY. Este tipo de comunicación tiene un mayor coste pero se obtiene un mayor control y gestión de la red. A esta red le llamaremos "activa" puesto que en cualquier momento se puede realizar una petición o modificación de cualquier parámetro y obtener la respuesta de ello.

- **One-to-one** o **One-to-many**
    - *One-to-one* se produce cuando dos dispositivos, transmisor y receptor, participan en la comunicación. Por su parte, *One-to many* es cuando el transmisor se comunica con un grupo de receptores.

Para una implementación sencilla, es mejor realizar una comunicación ONE-TO-ONE que ONE-TO-MANY. Una comunicación como esta última podría tener algún sentido en una aplicación similar a la anteriormente comentada de los incendios (o donde existan un gran número de sensores del mismo tipo que realicen la misma tarea) pero requiere también implementar funciones de control de grupos para enviar una comunicación a varios TIMs. Para un caso sencillo como es el que se está tratando de explicar, es mejor una comunicación ONE-TO-ONE y la forma de llevarlo a cabo es también más intuitiva.

- **Default-QoS** o **Special-QoS** (QoS, Quality of Service)
    - Cuando no se indique de otra manera, la calidad en la comunicación será *Default-QoS*, es decir, best-effort. Si por lo contrario se desea que cumpla algún parámetro será *Special-QoS.*

Salvo casos especiales donde se requieran un mayor control de un parámetro o que puedan existir errores de algún tipo debido a algún sensor el "quality of service" será el que se establezca por defecto. Este parámetro de comunicación no hay que dejarlo de lado pero tampoco es necesario realizar una implementación mayor que la que tiene. Es muy dependiente

de la aplicación y de las condiciones iniciales (tanto de la red como atmósfericas) que se tengan y en ocasiones puede ser mejor jugar con redundancia de mensajes que cargar computacionalmente un pequeño microprocesador.

- **Point-to-Point** o **Network**
    - La comunicación será *Point-to-Point* cuando el interfaz físico sea simple y para un dispositivo, y por su parte, se denominará *Network* cuando el medio sea compartido por varios dispositivos.

  Comunicaciones punto a punto hoy en día y cada vez más, se encuentran con mayor desuso y sólo pueden tener cierto sentido en comunicaciones cableadas. En una comunicación wireless en las que se va a trabajar con redes Zigbee, Bluetooth y otras muchas más las cuales permiten una creación y gestión de una red de dispositivos es muchísimo más ventajoso trabajar con en modo NETWORK porque si lo harías de la otra forma, tendrías que implementar en el NCAP una interfaz del módulo de comunicaciones para cada miembro de la red. Point-to-Point se ha quedado para implementaciones en las que se utilice el miembro IEEE 1451.2 como módulo de comunicaciones, puesto que este miembro (que fue el primero que salió de este tipo) que trabaja por medio de cable (Rs232…) o en situaciones en la que existan muy pocos nodos en la red.

Como se ha mencionado anteriormente, estas APIs mencionadas son simétricas. A continuación, se muestran los 3 patrones de comunicación contemplados por el estándar:

- Two-way/One-to-one: por ejemplo, un NCAP solicita el comando para leer una TEDS a un TIM específico y dicho TIM le contesta con el contenido correspondiente.

- One-way/One-to-one: por ejemplo, un TIM genera una medida periódica y la envía a un NCAP.
- One-way/One-to-many: por ejemplo, igual que caso anterior pero participando varios TIMs en la medida.

Para una implementación sencilla y típica el mejor patrón para llevar a la práctica es el primero: TWO-WAY/ONE-TO-ONE. Esta configuración es la clásica para aplicaciones, de hecho prácticamente la totalidad de artículos existentes hablando de comunicaciones en este sentido, ya que es la que permite la opción petición/respuesta de cada sensor. Teniendo además en cuenta, el enfoque que le hemos dado a nuestra aplicación, (comunicaciones inalámbricas mediante tecnologías Zigbee, 6LowPAN…) trabajaremos con microprocesadores de ultra bajo consumo, pero también estos dispositivos serán menos potentes que otros microprocesadores por lo que no conviene realizar comunicaciones con varios dispositivos simultáneamente es mejor, en dicha situación, apostar por una comunicación más completa con un único dispositivo.

Por otra parte, la comunicación aparte de ser Two-Way/One-to-one, será también DEFAULT-QoS y NETWORK. Como se ha explicado anteriormente, no es necesario establecer de antemano parámetros de "Special-Quality-of-Service" para aplicaciones normales, ya que los protocolos Zigbee, Bluetooth… tienen por sí mismos un alto nivel de fiabilidad y calidad, pudiendo controlar este factor en algunos de sus perfiles. Mucho más importante es la decisión de la implementación de un módulo de comunicaciones que trabaje con las interfaces NETWORK o con las POINT-TO-POINT. A día de hoy, quedaría descartada una implementación de un módulo de comunicaciones que trabaje con interfaces "point-to-point" salvo, como se ha explicado anteriormente, se trabaje con el miembro IEEE 1451.2 o en situaciones en las que se dispongan de una red con pocos TIMs (dos o tres). En una aplicación en la que se desee trabajar de forma inalámbrica (como la que nos acontece) o se vaya a manejar una gran cantidad de TIMs y sensores se implementará los interfaces "Network".

NOTA- *Las interfaces que contienen las funciones y métodos del Módulo de Comunicaciones las especifica de forma general IEEE 1451.0 pero se concreta su contenido y el valor de sus parámetros en el miembro IEEE 1451.X correspondiente, para este caso IEEE 1451.5. Más adelante, en el apartado de la implementación de IEEE 1451.5 se aclarará el Módulo de Comunicaciones.*

IEEE 1451.5

El Módulo de Comunicaciones

Las interfaces y funciones que componen el Módulo de Comunicaciones son definidas, de forma general, en IEEE 1451.0 donde se explica detalladamente el nombre y tipo de dato en que se debe implementar. No obstante es la correspondiente norma IEEE 1451.X (la responsable en crear la capa de tranporte) la encargada en profundizar en los valores y la forma que deben de tener las funciones y los métodos de cada interfaz.

Las funciones y métodos del Módulo de Comunicaciones pueden ser divididas en dos grandes grupos:

- Funciones y métodos destinados a los servicios:
  - Múltiples funciones y métodos son llevados a cabo llamando a la capa de IEEE 1451.0 donde se encuentra la función para realizarla. Esto ocurre para el caso de servicios y son la mayoría de ellas.

- Funciones y métodos destinados a intercambiar información con la capa física:
  - Son las funciones que tienen que ver con la tecnología que se utiliza para la comunicación NCAP-WTIM (funciones más relacionadas con la capa física). La implementación de todas ellas es obligatoria (y las que no, son muy recomendables)

9.5. TRANSDUCER SERVICES

La TransducerServices API proporciona la interfaz entre las aplicaciones que se ejecutan en el NCAP y las funciones definidas por el estándar IEEE 1451.0.

Las aplicaciones de medición y control interactúan con la capa del estándar IEEE 1451.0 por la Interfaz Transducer Service. A continuación se mostrarán algún ejemplo de la utilización de esta interfaz.

La petición de medida más simple y más común será la lectura de un valor de un TransducerChannel. Si suponemos que el TransducerChannel está siendo operado en ejecución inmediata *sampling mode*, el TransducerChannel realizará automáticamente la medida y devolverá el resultado.

El diagrama de secuencia en la figura 30 ilustra el flujo. La aplicación está a la izquierda, y se muestran los tiempos de ejcución. Los detalles de procesamiento por debajo de la API IEEE 1451.0 no están ilustrados:

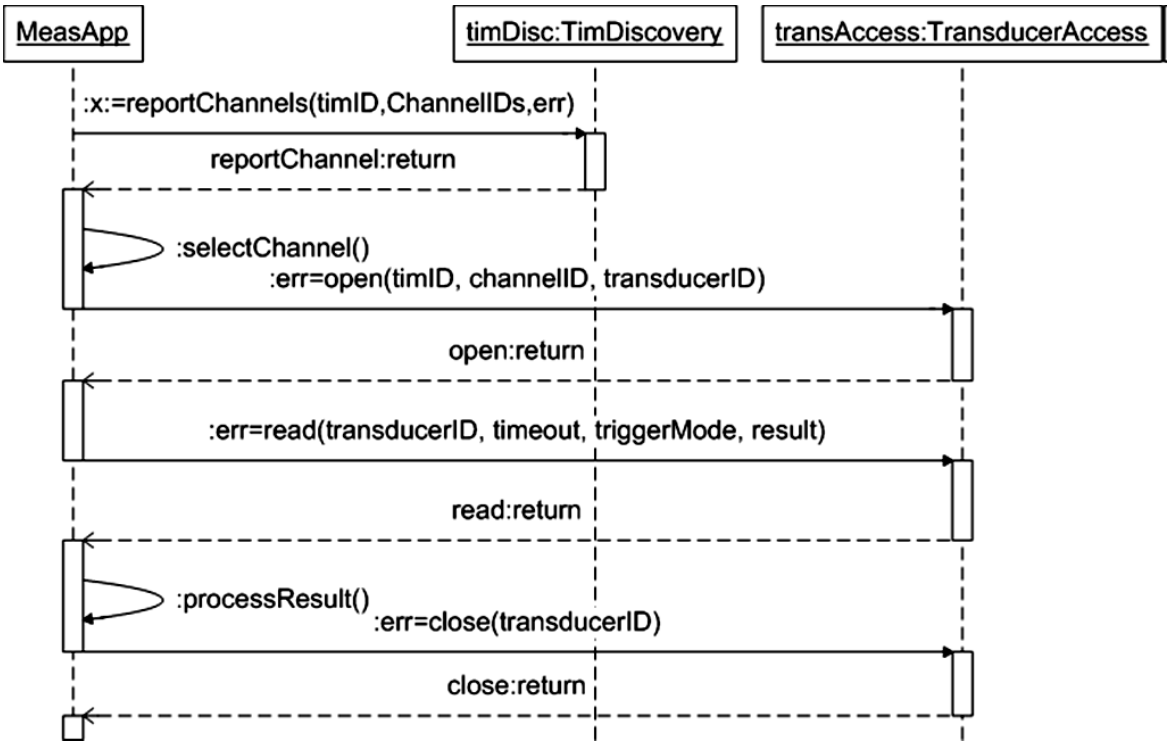


Ilustración 30: Diagrama de flujo del registro de un dispositivo en IEEE 1451.

La aplicación (con la etiqueta de objeto MeasApp) llama a las funciones y métodos correspondientes del TIMDiscovery para "descubrir" TIMs y TransducerChannels que se encuentren disponibles para asociarlos con el estándar.

Una vez que se ha realizado el registro de un canal (TransducerChannel), se llama a la función *open( )* de la interfaz TransducerAccess. Esta función devolverá el parámetro "transducerID" que se utilizará en las llamadas y peticiones posteriores.

Cuando la aplicación está lista para hacer una lectura de una medición, se llama a la función *read*( ). El valor de las nuevas mediciones serán devueltas en el parámetro de salida "result". El resultado es un objeto ArgumentArray, y la aplicación utiliza el método *get*( ) para recuperar los atributos de medición de interés. Por ejemplo, el atributo "value" tiene el valor de la medición, mientras que el atributo "timestamp" incluye el momento en que la medición se ha efectuado.

La aplicación puede controlar qué atributos se devuelven en el ArgumentArray mediante el método *configureAttributes*( ) que se encuentra en la interfaz TransducerManager (no mostrado en el anterior esquema). Por ejemplo, los atributos "units", "accurracy", "name" y "id" pueden ser habilitados. Esto obligará a la función *read*( ) que devuelva un ArgumentArray con estos atributos para un más completo "self describing" de los datos.

La aplicación puede llamar a la función *read*( ) con la frecuencia necesaria para adquirir múltiples lecturas. Cuando la aplicación ha terminado, se llama a la función *close*( ) para liberar los recursos.

El ejemplo anterior ilustra el caso más simple, donde las siguientes condiciones se deben mantener:

- Tan solo se puede acceder a un TransducerChannel a la vez.
- Operación-Bloqueante: la aplicación se bloquea hasta que se devuelve el resultado. El máximo tiempo que puede esperar la aplicación para que le contesten está determinado por el parámetro "timeout".

- La aplicación especifica el *sampling mode* deseado. En este ejemplo, se realiza el sampling mode de operación inmediata puesto que es el más común, es el recomendado en situaciones e implementaciones normales.

Por otra parte, también existe la posibilidad de realizar lecturas de datos de forma no-bloqueante pero como se ha comentado anteriormente, en este apartado de guía de implementación se va a explicar cómo realizar el caso más común y sencillo como el que se acaba de explicar (lectura bloqueante).

Envío de comandos

Como se señala en la cláusula 7 de la especificación IEEE 1451.0, la gran mayoría de los comandos, los *Standard Commands*, están definidos en la capa IEEE 1451.0 del estándar. En muchos casos, ciertos métodos específicos se proporcionan a través de la API para llevar a cabo la tarea de una forma más sencilla, por ejemplo, leer un TransducerChannel o TEDS. Uno de estos métodos es el mecanismo *sendCommand()*. Para acceder a las funciones que no están expuestas a través de métodos de "conveniencia", el método *sendCommand*( ) está disponible y se utiliza de la siguiente manera:

- Se llama a la función *open* ( ) para obtener un "transducerId", que es el identificador del WTIM o TransducerChannel.
- El "CommandID" se selecciona entre todos los posibles comandos que se enumeran en la cláusula 7. Este número UInt16 contiene el comando "clase", (8 bits más significativos) y el comando "función" (es el campo dentro de la clase del comando) representado por los 8 bits menos significativos.
- Todos los argumentos de entrada de los comandos específicos son empaquetados en el inArgs como “ArgumentArray”. La posición y el tipo de argumentos en la ArgumentArray se ajustará a lo que se requiere para el comando en cuestión.
- Se llama a la función *sendCommand* ().
- Los parámetros de salida se devuelven a través del “outArgs” ArgumentArray. Una vez más, la posición y tipo de argumentos en este ArgumentArray cumple con lo que es definido por el comando en cuestión.

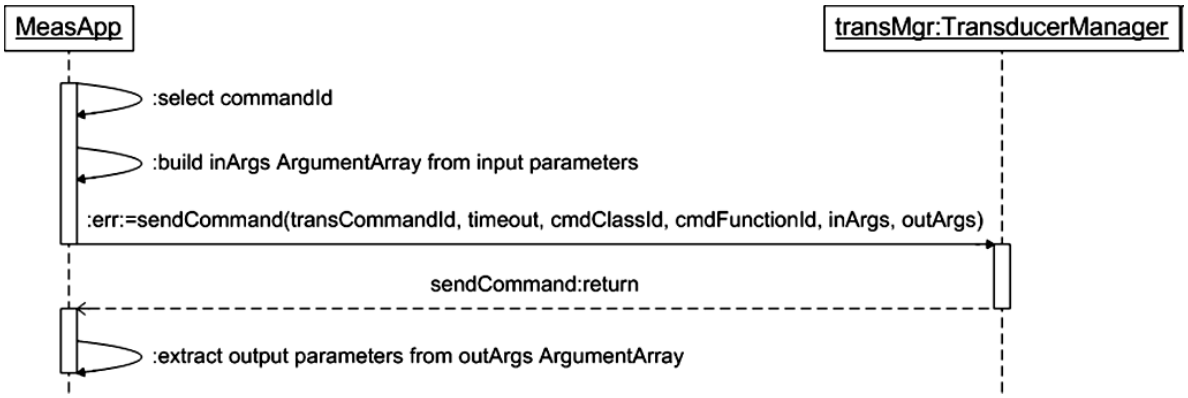


Ilustración 31: Diagrama de flujos de envío de comandos en IEEE 1451.0.

Diagrama de la secuencia de una operación de lectura

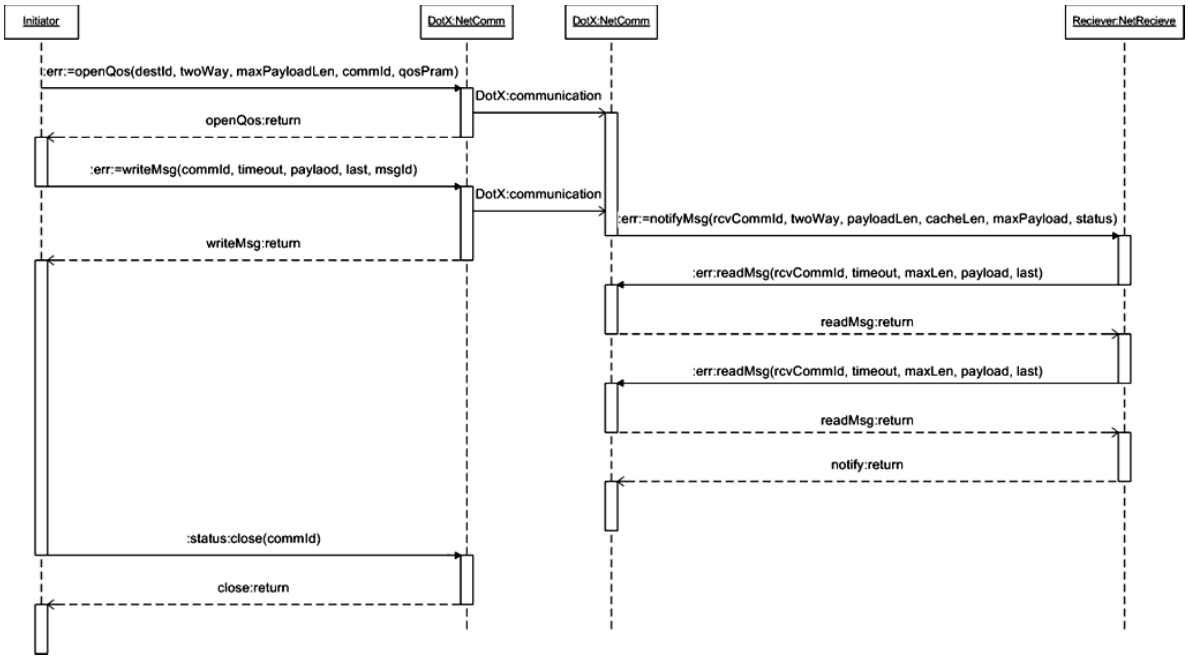


Ilustración 32: Diagrama de flujos de las funciones OPEN y lectura.

La aplicación está a la izquierda, y las capas IEEE 1451.X están en el centro. Notar que la llamada *open()* y el procesamiento en la parte del TIM está por encima de la capa IEEE 1451.0 que no se ilustra en este esquema. En este caso,

suponemos que la aplicación está realizando una lectura inmediata del *TransducerChannel 1* del TIM deseado. Una vez que la aplicación llama a la función *read()*, ocurre lo siguiente:

a) Se construye el apropiado comando para que el WTIM lea el TransducerChannel deseado con el sampling mode especificado (véase 7.1.2.4 de std. IEEE 1451.0) (operación de modo inmediato en este caso).

b) A continuación, se llama a la función *encode*( ) para construir el "*payload*" OctetArray del comando anterior. Al tratarse de un comando de lectura, el payload contendrá los siguientes 10 octetos

1) El identificador del TransducerChannel (0x0001 por ejemplo) como UInt16. Este es el número del TransducerChannel que se va a leer.

2) la clase de comando (0x02) como UInt8

3) "Function Command", función del comando, se trata de la operación dentro de la clase del apartado anterior, tiene longitud UInt8. El código de comando es " Read TransducerChannel data-set segment " (valor 1) en este caso.

4) Longitud de la información en octetos, 0x0004 como UInt16.

5) El DataSetOffset es el único argumento del comando que hay que enviar, y contiene 0x00000000 como UInt32, ya que el conjunto de datos contiene un solo valor.

6) El comando *write*( ) se invoca en la capa IEEE 1451.X para iniciar la transferencia del OctetArray hacia el WTIM. Esta comunicación se bloqueará mientras que el WTIM para hacer la medición. El tiempo máximo de espera es especificado por la aplicación.

7) Cuando el procesamiento en WTIM haya concluido, la capa IEEE 1451.X transferirá el resultado OctetArray de nuevo hacia el NCAP. Se llama a la función *notify*( ).

8) El procesamiento de las Notificaciones termina con el bloqueo anterior.

9) El método *read*( ) se invoca para obtener la respuesta OctetArray.

10) La función *decode*( ) es utilizada para convertir los OctetArray de nuevo a un ArgumentArray.

11) La función *convert* ( ) se utiliza para corregir los valores o adaptarlos al formato o unidades del receptor.

12) El formato de los ArgumentArray debe ser el correcto.

13) El ArgumentArray se devuelve con los parámetros "out" hacia la aplicación que había realizado la petición.

9.5.1. Funciones a implementar

A continuación se van enumerar las funciones y métodos de cada interfaz de esta API que típicamente hay que implementar para llevar a cabo

Interfaz	Descripción
TimDiscovery	Este interfaz contiene métodos para encontrar módulos de comunicaciones IEEE 1451.X, TIMs y TransducerChannels.
TransducerAccess	Este interfaz proporciona métodos cuando una aplicación desea tener acceso a sensores o actuadores TransducerChannels.
TransducerManager	Las aplicaciones que necesitan más control del acceso sobre TIMs usarán métodos de este interfaz.
TedsManager	Para usos o aplicaciones en las que se necesita leer y/o escribir TEDS, gestión de TEDS.
CommManager	Se necesita acceso al módulo de comunicaciones del dispositivo.
AppCallback	Cuando ciertos usos o aplicaciones necesiten de opciones avanzadas. IMPORTANTE: en este estándar no se define ningún AppCallback. Está pensado para poder personalizar la red y para futuras modificaciones si hiciera falta.

Tabla 40: Interfaces que componen el Transducer Service API de IEEE 1451.0.

A continuación, se irán mostrando las funciones que deben de ser implementadas dentro de cada interfaz.

Interfaz Tim Discovery

Esta interfaz es fundamental para realizar un control y una gestión satisfactoria de nuestra red de sensores.

Se llevará a cabo la implementación de las tres funciones o métodos que tiene este interfaz: *reportCommModule*( ), *reportTims*( ), *reportChannels*().

reportCommModule

```
IDL: Args::UInt16 reportCommModule( out Args::UInt8Array moduleIds);
```

Esta función permite conocer al usuario remoto los módulos de comunicaciones que tiene asociados la correspondiente capa 1451.0. Tener en cuenta que esta petición puede realizarse de dispositivo NCAP pero también de un determinado TIM.

La función tiene que tener el nombre y parámetros de entrada/salida tal y como se muestra.

reportTims

```
IDL: Args::UInt16 reportTims(
in Args::UInt8 moduleId,
out Args::UInt16Array timIds);
```

Esta función permite conocer al usuario remoto los TIMs que tiene asociados un módulo de comunicaciones especificado mediante el parámetro de entrada "moduleId" de un determinado NCAP.

reportChannels

```
IDL: Args::UInt16 reportChannels(
in Args::UInt16 timId,
out Args::UInt16Array channelIds,
out Args::StringArray names);
```

Esta función permite conocer al usuario remoto los canales de comunicaciones (channels) que tiene asociados un TIM especificado mediante el parámetro de entrada "timId".

Interfaz TransducerAccess

Se implementarán las siguientes funciones o métodos: open( ) readData( ) writeData( ) close( ) cancel( )

open

```
IDL: Args::UInt16 open(
in Args::UInt16 timId,
in Args::UInt16 channelId,
out Args::UInt16 transCommId);
```

Una vez que ya están registrados conforme al estándar tanto un TIM como un Channel de dicho TIM, esta función permite comenzar a intercambiar datos, al devolver el parámetro "transCommId" que será usado para cuando se quiera acceder o recuperar datos de un determinado "channel".

readData

```
IDL: Args::UInt16 readData(
in Args::UInt16 transCommId,
in Args::TimeDuration timeout,
in Args::UInt8 SamplingMode,
out Args::ArgumentArray result);
```

Este método realiza una lectura bloqueante de un especificado TransducerChannel.

writeData

```
IDL: Args::UInt16 writeData(
in Args::UInt16 transCommId,
in Args::TimeDuration timeout,
in Args::UInt8 SamplingMode,
in Args::ArgumentArray value);
```

Este método realiza una escritura bloqueante de un especificado TransducerChannel.

close

```
IDL: Args::UInt16 close( in Args::UInt16 transCommId);
```

Esta función es la complementaria de la función "open" ya que convierte un canal de comunicación de "open" a "registrado", dejando el parámetro "transCommId" como inválido para realizar comunicaciones.

cancel

```
IDL: Args::UInt16 cancel( in Args::UInt16 operationId);
```

Este método cancela una operación bloqueante (lectura, escritura o medidas en el stream). La llamada se realiza a través del estado código de error CANCEL. Aunque no se trata de una función indispensable para el funcionamiento del sistema, se ha decidido su implementación puesto que es muy sencilla de implementar y porque como se ha mostrado anteriormente se ha apostado por una implementación que realiza funciones bloqueantes por lo que sería muy positivo para nuestro sistema tener una pequeña función que nos permita cancelar una de estas funciones.

Interfaz TransducerManager

Se implementarán las siguientes funciones o métodos: lock ( ), unlock ( ) , reportLocks ( ), sendCommand ( ), configureAttributes ( ), trigger ( ), clear ( ) y unregisterStatusChange ( ):

lock

```
IDL: Args::UInt16 lock(
in Args::UInt16 transCommId,
in Args::TimeDuration timeout)
```

Este método bloqueará un TIM / TransducerChannels específico representado por el transCommId. De esta manera se evitará que otras aplicaciones tengan acceso a esos recursos.

unlock

```
IDL: Args::UInt16 unlock( in Args::UInt16 transCommId);
```

Este método desbloqueará un TIM / TransducerChannels específico representado por el transCommId. De esta manera se vuelve a permitir que otras aplicaciones tengan acceso a esos recursos.

reportLocks

```
IDL: Args::UInt16 unlock(out Args::UInt16Array transCommIds);
```

Esta function devuelve una lista con los "transCommIds" que se encuentran bajo bloqueo.

sendCommand

Importante función en el envío de comandos, sobre todo a la hora en la que se creen nuevos comandos.

```
IDL: Args::UInt16 sendCommand(
in Args::UInt16 transCommId,
in Args::TimeDuration timeout,
in Args::UInt8 cmdClassId,
in Args::UInt8 cmdFunctionId,
in Args::ArgumentArray inArgs,
out Args::ArgumentArray outArgs);
```

configureAttributes

```
IDL: Args::UInt16 configureAttributes(
in Args::UInt16 transCommId,
in Args::StringArray attributeNames);
```

trigger

```
IDL: Args::UInt16 trigger(
in Args::UInt16 transCommId,
in Args::TimeInstance triggerTime,
in Args::TimeDuration timeout,
in Args::UInt16 SamplingMode);
```

clear

```
IDL: Args::UInt16 clear(
in Args::UInt16 transCommId,
in Args::TimeDuration timeout,
in Args::UInt8 clearMode);
```

Interfaz TedsManager

Se implementarán las siguientes funciones o métodos: readTEDS( ) y writeTEDS( ).

readTEDS

Este método realiza la lectura de una TEDS especificada de la caché TEDS. Si la TEDS no está disponible en la caché, se leerá del TIM. La información TEDS se devuelve en un *ArgumentArray*.

```
IDL: Args::UInt16 readTeds(
in Args::UInt16 transCommId,
in Args::TimeDuration timeout,
in Args::UInt8 tedsType,
out Args::ArgumentArray teds);
```

writeTEDS

Este método realiza la escritura de una TEDS en el TIM. La caché TEDS también se actualiza y la almacena si la escritura se realiza correctamente. La información proporcionada, está codificada en un *ArgumentArray*. Será convertida internamente a la forma correcta "tupla" (terna) y será transferido al TIM en un OctetArray.

```
IDL: Args::UInt16 writeTeds(
in Args::UInt16 transCommId,
in Args::TimeDuration timeout,
in Args::UInt8 tedsType,
in Args::ArgumentArray teds);
```

Interfaz CommManager

Esta interfaz contiene tan solo un método, el "getCommModule( )".

getCommModule

```
IDL: Args::UInt16 getCommModule(
in Args::UInt8 moduleId,
out ModuleCommunication::Comm commObject,
out Args::UInt8 type,
out Args::UInt8 technologyId);
```

Interfaz AppCallback

Como se ha explicado anteriormente, el estándar IEEE 1451.0 no especifica ninguna función o método de esta interfaz y en principio no tiene porque existir. Su desarrollo estará en manos del fabricante o del personal encargado de llevar a cabo

la implementación y dependerá de las características del hardware o según funcionalidades requeridas para el software. Son implementadas para realizar algún requerimiento de la aplicación o como paso intermedio para una funcionalidad no contemplada por el estándar.

9.6.EJEMPLO. IMPLEMENTACIÓN EN EZ430-RF2480

A continuación, se va explicar la implementación que se ha llevado a cabo del estándar IEEE 1451 en los dispositivos Z-Accel Demonstration Kit (ez430-RF2480) de Texas Instruments.

El eZ430-RF2480 Demo Kit es una herramienta (demonstration kit) inalámbrica basada en el microprocesador 430 (con conectividad USB) que incluye todo el hardware y el software para evaluar el procesador de red ZigBee mediante CC2480 (a 2,4 GHz) y el microprocesador MSP430F2274 para realizar una comunicación inalámbrica mediante un procesador con protocolo Zigbee/802.15.4. Este kit de demostración tiene todo lo necesario para comprender y evaluar plenamente las capacidades de CC2480 en un corto período de tiempo de forma sencilla.

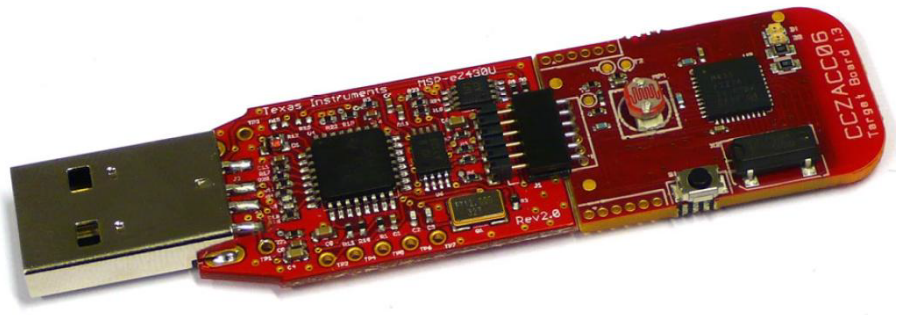


Ilustración 33: Dispositivo del Kit Z430-RF2480 formado por una tableta conectada al dispositivo USB.

El eZ430-RF2480 Demo Kit utiliza una versión gratuita de Kickstart del entorno de desarrollo integrado IAR Embedded Workbench (IDE) para escribir, descargar y depurar la aplicación. A continuación se muestran las características principales del MSP430:

Características	**MSP430F2274**
Frequency(MHz)	16
Flash	32 KB
RAM	1 KB
GPIO	32
Pin/Package	38TSSOP, 40VQFN
ADC	10-Bit SAR
Other Integrated Peripherals	2 Op Amp
Interface	1 USCI_A (UART/LIN/IrDA/SPI),1 USCI_B (I2C/SPI)
Timers	1 Watchdog/Interval, 2 16-bit (3CCR)

Tabla 41: Características generales del microprocesador MSP430F2274.

Por defecto, el microprocesador MSP430F2274 que viene integrado en la placa del "starter kit", viene con una aplicación ZASA y mediante la herramienta IAR Workbench se puede realizar debugging o modificar el programa para personalizar la aplicación o para que realice otra tarea distinta. El lenguaje de programación utilizado por la herramienta para el desarrollo de la aplicación es C. A continuación, entraremos más a fondo en el funcionamiento del Kit.

9.6.1. Acerca del Kit eZ430-RF2480

Antes de entrar en detalles definiremos los tres tipos distintos de dispositivos ZigBee/802.15.4 en lo que a arquitectura hardware se refiere:

- Arquitectura basada en un transceptor 802.15.4 y un Microprocesador. En este tipo de dispositivos el transceptor se ocupa de la parte radio (capa física 802.15.4) mientras que el microprocesador se encarga de gestionar la capa MAC, de red y de aplicación.
- Arquitectura basada en un procesador ZigBee/802.15.4 y un Microprocesador. Ahora, el procesador ZigBee/802.15.4 gestiona tanto la capa física como las capas MAC y de red, dejando al microprocesador encargado únicamente de la parte de aplicación. Estos sistemas son más simples, pero menos flexibles, de cara al desarrollador puesto que éste sólo se debe ocupar de la aplicación, dejando que el procesador ZigBee/802.15.4 gestione el resto de capas.
- Arquitectura basada en un dispositivo completamente integrado (System-on-chip, SoC). En este último caso, tanto el transceptor como el microprocesador están integrados en un único chip.

El kit de desarrollo eZ430-RF2480 pertenece al segundo tipo de los mencionados anteriormente, se trata de un procesador ZigBee/802.15.4 más microprocesador. El procesador ZigBee/802.15.4 es un CC2480 y, como hemos dicho, realiza las tareas de la capa física, MAC y de red. Ambos dispositivos están conectados por un bus SPI. En la figura podemos ver una representación de la distribución de las capas del protocolo ZigBee/802.15.4 entre CC2480 y MSP430 A esta combinación de procesador ZigBee/802.15.4 y del protocolo Z-Stack (CC2480), TI la denomina Z-Accel. El objetivo es hacer que el desarrollador se centre en la creación de su aplicación dejando que el dispositivo Z-Accel se encargue de la parte ZigBee/802.15.4.

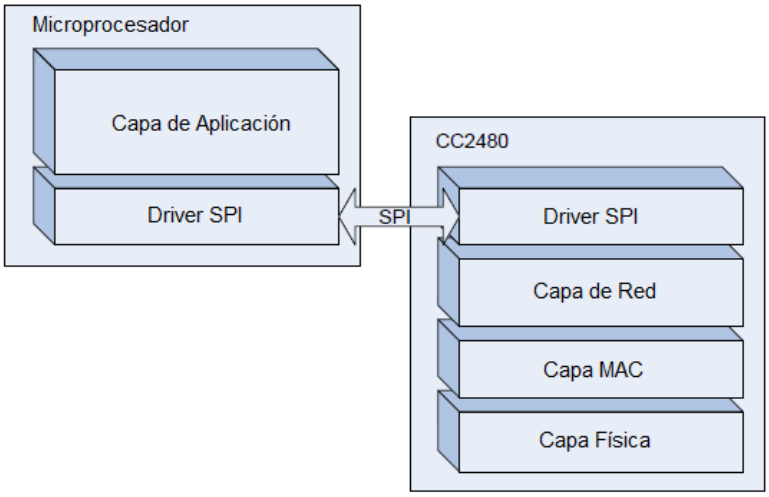


Ilustración 34: Comunicación entre MSP430 y CC2480.

La aplicación que, por defecto, viene instalada en el MSP430 se denomina ZASA (ZigBee Accelerator Sample Application) y pretende ser un ejemplo simple de una aplicación ZigBee.

El kit eZ430-RF2480 está compuesto por tres "tabletas" iguales, un conector USB y dos conectores para baterías AAA. La tableta, aparte de contener el procesadorCC2480 y el microprocesador MSP430 también está compuesta por un sensor de luz, un botón, dos diodos LED (uno verde y otro rojo) y por una antena de tipo chip.

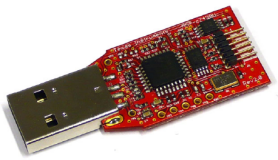


Ilustración 35: Conector USB.

Ilustración 36: Conector para baterias AAA.

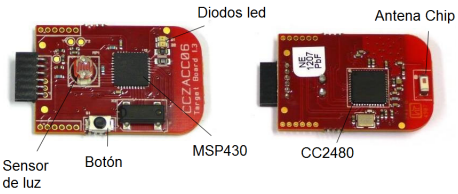


Ilustración 37: Tarjetas (MSP430, CC2480, diodos LED...).

En la siguiente figura vemos como las diferentes partes se conectan entre sí. Como vemos, con un kit disponemos de hasta tres nodos ZigBee/802.15.4 para formar nuestra red. Estos nodos son de tipo FFD puesto que pueden funcionar

como coordinador, router o dispositivo final. Dos dispositivos están alimentados por baterías y un tercero lo está a través de la conexión USB a un PC. Gracias al dispositivo conectado al PC, podemos visualizar diferentes parámetros de la red.

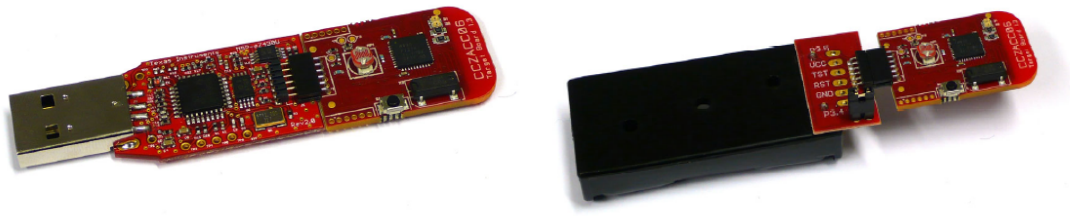


Ilustración 38: Dispositivos que utilizaremos como coordinador (izqda.) y como nodo final (drcha).

Cuando utilizamos los dispositivos con ZASA como aplicación, es en el momento de la inicialización cuando definimos la funcionalidad del dispositivo. Así, una configuración básica sería configurar el primer dispositivo como coordinador, el siguiente como router, y el último de ellos como dispositivo final. Una vez los dispositivos han sido configurados, todos ellos comienzan a enviar de forma periódica datos al coordinador.

Como hemos dicho, el CC2480 es un procesador ZigBee/802.15.4 que abarca las capas física, MAC y de red. Posee una conexión SPI/UART para poder conectarse con cualquier microprocesador. La implementación ZigBee de TI, Z-Stack, está cargada en el CC2480. CC2480 cumple con el estándar ZigBee-2006

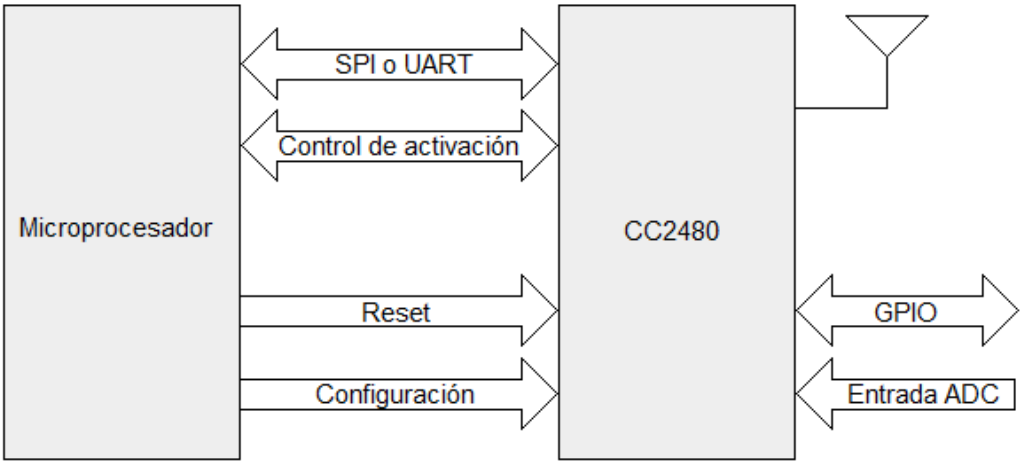


Ilustración 39: Interfaz SPI/UART entre el microprocesador y el CC2480.

La interfaz SPI/UART es el bus que permite el intercambio de datos entre CC2480 y el microprocesador. SPI es un bus serie de transmisión síncrono. UART es un bus asíncrono que convierte los bits de paralelo a serie antes de enviarlos.

Simple API es una versión reducida de la interfaz API que proporciona Texas Instrument para Z-Stack. La interfaz Z-Stack API está pensada para dispositivos en los que la pila de protocolos está cargada en el procesador (sección 3.1.1). Entre otras, Z-Stack API está compuesta por una serie de primitivas con las que manejar tanto la capa de red como la subcapa APS. Como hemos dicho, CC2480 abarca las capas física, MAC y de red. Por lo tanto no es necesario disponer de todas las primitivas de lasque está compuesta Z-Stack API. La solución es esta versión reducida conocida como simple API.

El CC2480 permite a los dispositivos finales activar el modo de bajo consumo cuando no están enviando información. Por el contrario, tanto el coordinador como los routers deben estar siempre despiertos (deben tener el sistema radio encendido y con el procesador CC2480 activo).

El kit eZ430-RF2480 hace uso de un microprocesador MSP430 como sistema de control del procesadorCC2480. Texas Instruments dispone de varios modelos dentro de la familia MSP430. La diferencia básica entre ellos es la cantidad de memoria Flash (ROM) y RAM de que disponen. El modelo exacto de microprocesador que usa este kit es MSP430F2274.

El MSP430 es un microprocesador RISC (Reduced Instruction Set Computer) de 16 bits. Los procesadores RISC se basan en un número reducido de instrucciones que, para este caso, es de 27. Que el procesador sea de 16 bits implica que todos los registros y buses internos de comunicaciones estén formados por 16 bits.

La familia MSP430 está orientada a conseguir un bajo consumo. Es por ello que resulta idónea para el control de transceptores y procesadores ZigBee/802.15.4. Siempre que el microprocesador está esperando una interrupción es de vital importancia apagar los relojes que no estemos usando si queremos disminuir el consumo.

Aplicación de ejemplo ZASA

ZASA, o ZigBee Accelerator Sample Application por sus siglas en inglés, es la aplicación que viene cargada por defecto en el microcontrolador MSP430. Su objetivo es demostrar de una manera simple cómo funciona una red basada en tecnología ZigBee/802.15.4.

Básicamente lo que hace esta aplicación es centralizar y monitorizar la lectura de unos sensores de temperatura y voltaje. ZASA nos proporciona dos puntos de acceso a nivel de aplicación: receptor y fuente. La función de receptor sólo la puede realizar el coordinador. Su tarea será recibir las lecturas de temperatura y voltaje que las fuentes le proporcionan. La función de fuente la realizan los routers y los nodos finales. Por defecto, cada 10 segundos la fuente envía los datos al sumidero. Si la fuente, bien sea un router o un *endpoint*, está directamente conectada al coordinador el envío únicamente debe efectuar un salto. Si por el contrario la fuente está conectada a un router, éste tendrá que recibir los datos y encaminarlos hacia el coordinador por la ruta correspondiente.

Como hemos dicho, el procesador CC2480 está preprogramado para realizar las tareas de las capas inferiores del protocolo ZigBee por lo que no tenemos acceso a

él. Por contra, podemos modificar la aplicación cargada en el MSP430 y situada en lo alto de la pila de protocolos (ZASA en este caso).El mecanismo para modificar dicha aplicación, como hemos mencionado anteriormente, es a través del entorno de desarrollo IAR Embedded Workbench.

Funcionamiento

El kit de desarrollo eZ430-RF2480 proporciona tres dispositivos FFD por lo que, potencialmente, todos pueden ser configurados como coordinador, router o dispositivo final. Por supuesto, al programa runa aplicación ZigBee podemos definir de antemano de qué tipo será nuestro dispositivo. Sin embargo ZASA nos da la oportunidad de elegir si un dispositivo será coordinador, router o dispositivo final (nodo final) en pleno proceso de ejecución. Esto se consigue gracias al uso del pulsador que incorpora la tableta. Cuando encendemos un dispositivo éste queda a la espera. El usuario puede reconocer este estado porque ambos LED, tanto el rojo como el verde, parpadean una vez por segundo. En el momento que el usuario presiona una vez el botón se inicia una ventana de espera que dura dos segundos. Si se vuelve a pulsar una segunda vez el botón dentro de ese margen de tiempo, ZASA interpretará que el dispositivo debe comportarse como un dispositivo final. Si dentro de esos dos segundos no volvemos a presionar el botón, el dispositivo se configurará como router e intentará conectarse a un coordinador. Si no encuentra un coordinador, el propio dispositivo funcionará como coordinador e inicializará la red.

Tipo dispositivo	LED rojo	LED verde
Sin configurar	Parpadeando	Parpadeando
Coordinador	Permiso de conexión:encendido Permiso de conexión: parpadeando	Datos recibidos: un parpadeo
Router	Envío de datos: un parpadeo	Permiso de conexión: Encendido Sin permiso de conexión: parpadeando
End Device	Envío de datos: un parpadeo	Parpadeando

Ilustración 40: Estado de los diodos LED en la aplicación ZASA.

Como ya sabemos, el kit eZ430-RF2480 viene incorporado con un adaptador USB. Dicho adaptador proporciona una conexión UART entre el MSP430 y un PC. Idealmente el coordinador debe ir conectado a este adaptador lo que permitirá la monitorización en un PC los datos enviados por todos los dispositivos que funcionen como fuente. El programa utilizado para dicha monitorización que nos proporciona Texas Instruments se llama Sensor Monitor. Aparte de la posibilidad de monitorizar, también se puede aprovechar esta conexión UART para comunicarnos de forma más general con el microprocesador en tiempo de ejecución.

Estructuración del código

ZASA está escrito en lenguaje C y está pensado para ser compilado y cargado en el MSP430 usando el entorno de desarrollo IAR Embedded Workbench.

El código de ZASA se divide en los siguientes módulos:

- APP (Aplicación). Implementa la funcionalidad a más alto nivel. Se encarga de hacer que el dispositivo funcione según lo descrito en el punto anterior.
- HAL (Hardware Abstraction Layer). Esta parte del código se encarga de controlar al MSP430.Gestiona las conexiones SPI y UART, los relojes software, modos de funcionamiento (activo, LPM) y dispositivos externos conectados al MSP430 como el pulsador o los LED.
- MT (Monitor Test). Realiza las funciones necesarias para poder comunicar el MSP430 con el PC a través de una conexión UART. MT envía una copia de todos los comandos intercambiados entre MSP430 y CC2480 a través de la conexión UART. Así, podremos ver en nuestro PC todos los comandos que se intercambian dichos dispositivos. Además de esto el módulo MT también nos permite ordenar al MSP430, desde el PC, el envío de un comando determinado al CC2480. Los programas Sensor Monitor y Z-Tool se comunican con el microprocesador gracias a esta parte del código.
- SAPI (Simple API). Gestiona la interfaz Simple API vista en la sección 3.1.2 de este mismo capítulo.
- ZACCEL (Z-Accel). Controla el intercambio de comandos entre MSP430 y CC2480 actuando como maestro y esclavo, respectivamente, a través de la conexión SPI.

Sensor Monitor

*Sensor Monitor*es una aplicación diseñada para trabajar conjuntamente con los nodos del kit eZ430-RF2480 cuando estos ejecutan la aplicación ZASA. Esta aplicación para PC nos permite monitorizar los datos recibidos por el coordinador y procedentes de las fuentes (routers y dispositivos finales).Es una aplicación gratuita que nos proporciona Texas Instruments como herramienta para la monitorización.

*Sensor Monitor*es una aplicación muy simple que puede resultar muy útil para familiarizarse con el kit eZ430-RF2480.

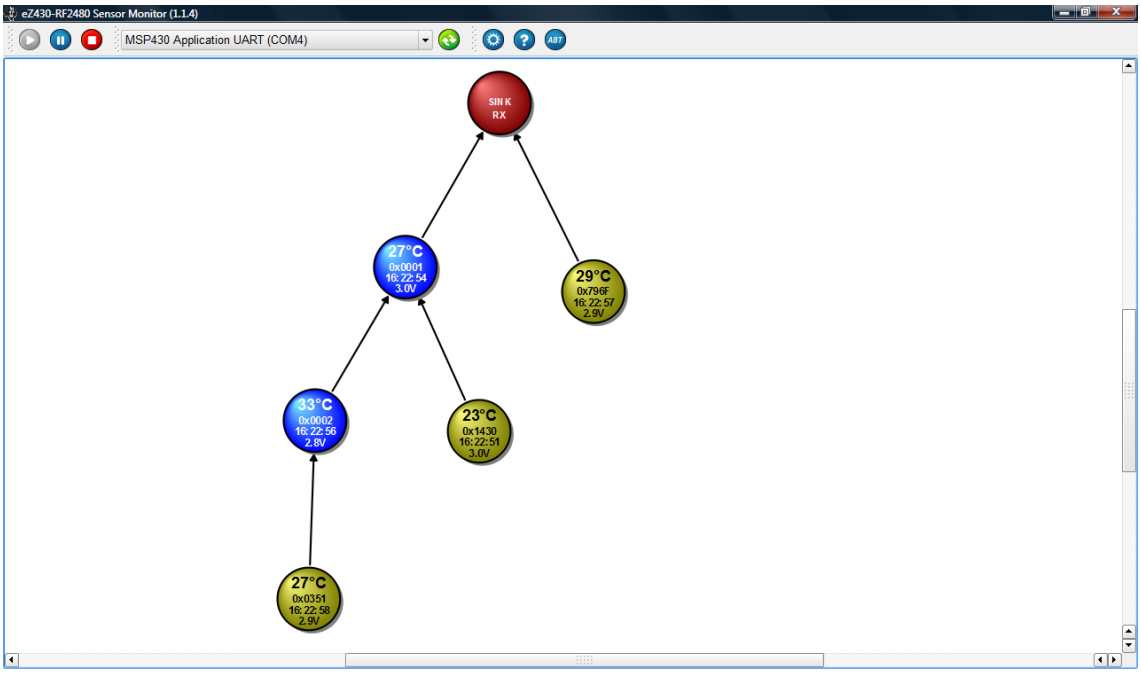


Ilustración 41: Captura de pantalla de la aplicación Sensor Monitor.

En este caso, la red está formada por 6 nodos (hemos utilizado dos kits eZ430-RF2480). El coordinador se representa en color rojo, los routers en azul y los dispositivos finales en dorado. Se ha configurado el dispositivo que utiliza el adaptador USB como coordinador de red. Este dispositivo será el que se comunique con *Sensor Monitor*a través de la conexión UART entre MSP430 y PC. El módulo MT del código de ZASA es el encargado de gestionar el intercambio de información entre el microprocesador y el PC. Vemos que Sensor Monitor nos permite visualizar

los datos que recibe el coordinador. Así, podemos ver cuál es la temperatura y el voltaje que lee cada nodo, su dirección corta y la topología general de la red.

IAR Embedded Workbench

IAR Embedded Workbench es un entorno de desarrollo especialmente pensado para la familia de microprocesadores MSP430 de Texas Instruments. Con esta herramienta compilamos el código (escrito en lenguaje C) y posteriormente lo cargamos en el microprocesador. Texas Instruments nos ofrece el código fuente de ZASA de manera gratuita por lo que, usando este entorno de desarrollo, podemos modificarlo y cargarlo en el microprocesador cuantas veces veamos oportuno.

Como cualquier otro entorno de desarrollo, IAR Embedded Workbench está compuesto principalmente por la ventana donde escribir el código fuente, el espacio de trabajo y la ventana de depuración.

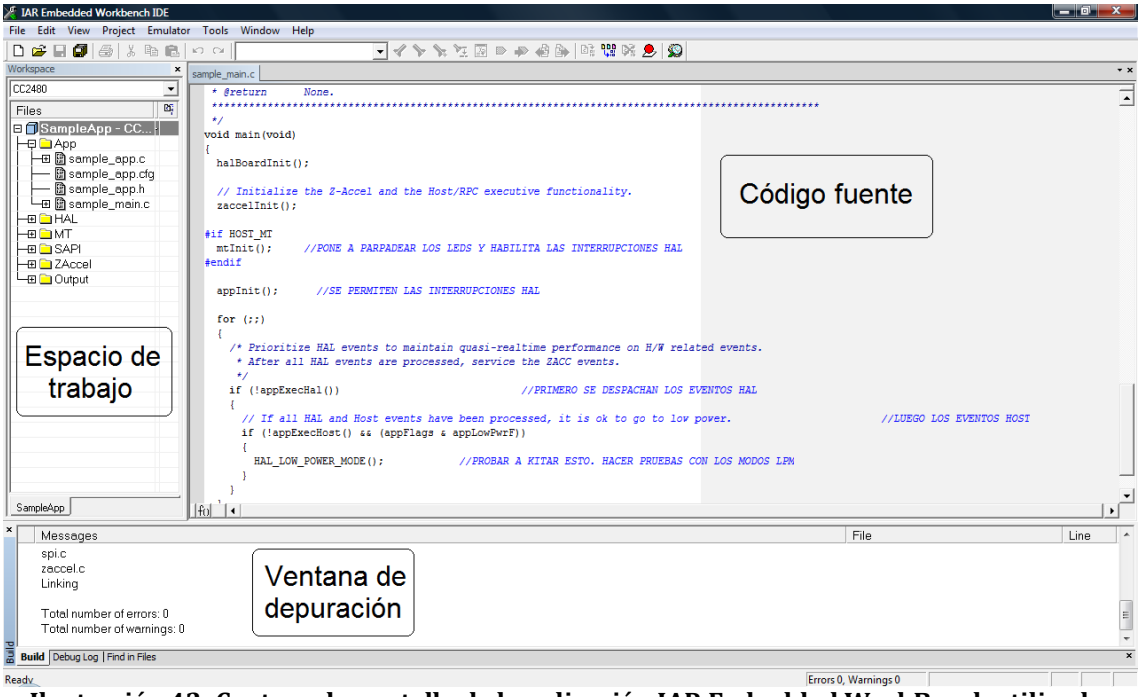


Ilustración 42: Captura de pantalla de la aplicación IAR Embedded WorkBench utilizada para modificar el programa ZASA.

9.6.2. La Implementación

Para la realización de la implementación del estándar IEEE 1451 en este kit de Texas Instruments se han tomado una serie de decisiones con el fin de respetar al máximo la filosofía del estándar. Hay que decir, que la elección de este kit como candidato para soportar el estándar se produjo antes de conocer cómo funcionaba el estándar y sin saber a ciencia cierta si se podía implementar en estos dispositivos y si iban a existir limitaciones al realizarla.

Como se ha dicho, este kit trabaja bajo tecnología inalámbrica IEEE 802.15.4/Zigbee por lo que está claro que el miembro IEEE 1451.5 del estándar se tiene que implementar puesto que es el que trata las redes de sensores inalámbricos (Zigbee, Bluetooth, Wi-Fi y 6LowPAN). Este miembro del estándar especifica la capa de transporte entre los dispositivos NCAP y WTIM (TIM inalámbrico), es decir el Módulo de Comunicaciones. En consecuencia, según la norma, para implementar IEEE 1451.5 hay que implementar obligatoriamente IEEE 1451.0. Este miembro especifica los comandos, la estructura de los mensajes, las TEDS (menos la PHY TEDS) y los servicios del sistema, mientras que IEEE 1451.5 especifica la forma de comunicación entre los dispositivos así como funcionalidades de la red de sensores.

La siguiente decisión en tomar es establecer qué dispositivos van a proporcionar las funcionalidades del NCAP y WTIM.

NCAP

El NCAP está formado por el dispositivo ez430-RF2480 con conexión USB y un PC, es decir, va a ser el coordinador de la red Zigbee. El motivo de incluir el PC como parte del dispositivo con funcionalidad NCAP es porque éste debe tener acceso a la red para poder conectarse con un cliente remoto para responder a sus peticiones y/o cambiar el estado de los actuadores a través de los comandos, conectarse a otra red de IEEE1451...

El NCAP es un dispositivo entre los WTIM y la red cableada y realiza las conexiones a múltiples WTIMs con la misma capa física. En nuestra aplicación la capa física es IEEE 802.15.4. EL NCAP puede iniciar la búsqueda para el registro y la asociación del nodo inalámbrico (WTIM) en cualquier momento.

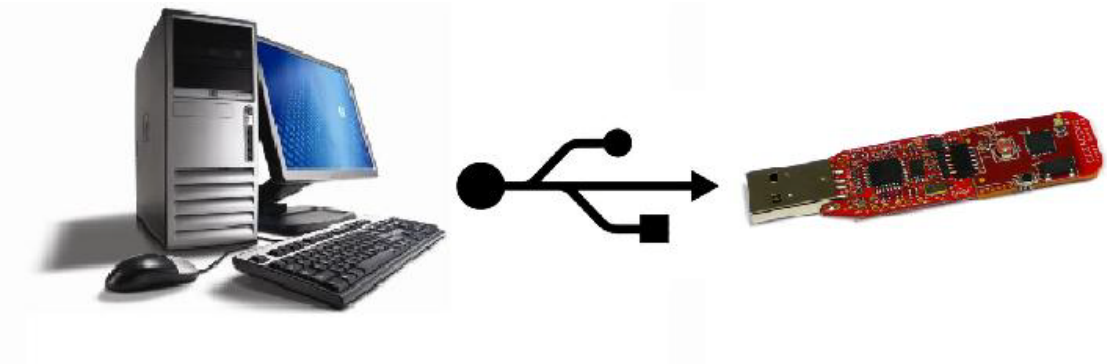


Ilustración 43: conjuntos de dispositivos que realizan las funcionalidades del NCAP.

NCAP: formado por conexión del coordinador Zigbee y el PC (encargado de la red de usuario)

WTIM

Las funcionalidades del WTIM son cubiertas por el dispositivo *"end_device"* (como se muestra en la siguiente figura) del kit eZ430-RF2480.

Ilustración 44: Dispositivo que realiza la funcionalidad del WTIM.

Una vez que ha quedado claro los dispositivos que van a realizar las funcionalidades del NCAP y WTIM, ha llegado un punto crucial para la implementación, el estándar IEEE 1451 debe de convivir con la aplicación ZASA.

Resulta complejo llevar a cabo la implementación del estándar en este kit de Texas Instruments, puesto que durante el estudio del estándar se explica todas las funcionalidades, la arquitectura, tipos de datos, etc., que se debe realizar desde un

punto de vista en el que el estándar IEEE 1451 es la aplicación principal, es decir, como si fuera el programa principal. El problema es que hay que modificar la aplicación ZASA que viene por defecto en el MSP430 para introducir el código del estándar. No se puede eliminar por completo la aplicación ZASA porque tiene interfaces con el procesador CC2480 que son vitales para la aplicación (controla nivel físico, MAC y de red), por lo que para realizar la implementación se respetará la estructura de ZASA y se introducirá el código del estándar donde sea oportuno. La arquitectura de IEEE 1451.0 como se había visto quedará sólo de forma conceptual.

Como se ha comentado anteriormente, el MSP430 es en el que se pueden realizar los cambios en la aplicación ZASA. Este microprocesador tiene 32KB de memoria ROM y 1KB de memoria RAM. La aplicación ZASA ocupa 4200 bytes de memoria (Flash) y 810 bytes de memoria RAM.

Los mensajes que lleguen al NCAP provenientes desde la red de usuario deben de ser tratados por la interfaz HTTP API de IEEE 1451.0 que debería de ser implementada en el PC, una vez tratados los mensajes en esta API los comandos son enviados a la Transducer Service API que debe de ser implementada en el MSP430.

No obstante, la HTTP API se considera el último paso de la implementación y se realizaría si el resto ha sido convenientemente implementado, mientras tanto, simularemos que los mensajes que recibe de la red usuario el dispositivo eZ430-RF2480 (conectado a través de USB con el PC), son enviados/recibidos con el software *Hyperterminal*. Con este software "interfaz" realizaremos las peticiones y visualizaremos las respuestas que hacemos sobre el dispositivo. El *Hyperteminal* es una herramienta fácil y sencilla de manejar que se puede utilizar para interactuar con el NCAP hasta que se implemente la parte del estándar que controla la comunicación desde el NCAP hacia la red de usuario. El Hyperteminal nos permite mandar/recibir datos desde el dispositivo eZ430-RF2480 conectado por USB y PC utilizando la UART del MSP430.

Ilustración 45: Comunicación a través de la UAR del dispositivo Zigbee con el Hyperterminal.

El *Hyperterminal* se comporta como una interfaz que nos permite comunicarnos con el dispositivo cuando está en pleno funcionamiento.

Limitación. Memoria RAM

Como se ha comentado anteriormente, la aplicación ZASA hace uso de 810 bytes por lo que existe un problema con la memoria RAM, ya que sólo quedan libres 190 bytes para realizar la implementación del estándar.

Se ha tratado de eliminar parte del código de la aplicación ZASA para ganar más memoria libre pero tan sólo se ha podido añadir unos pocos bytes a la memoria RAM.

Típicamente, las cuatro TEDS obligatorias (Meta-TEDS, TransducerChannel TEDS, User´s TEDS y PHY TEDS) ocupan, como se ha comentado en el apartado de las TEDS, en torno 200 bytes de los cuales, 65 bytes se pueden almacenar en memoria ROM pero el resto (135 bytes) irá alojada en memoria RAM. Como se observa sólo las TEDS ocupan más de la mitad de la memoria RAM disponible sin tener en cuenta las funciones y métodos para realizar los comandos del estándar, por lo que no se va a poder implementar de forma completa el estándar en estos dispositivos.

Por este motivo, se comenzará a realizar la implementación por las funciones y métodos de las interfaces que controlan el registro de los dispositivos con el estándar.

A la hora de realizar la implementación recordamos las características que se va a cumplir la comunicación: One-to-One, Network, Two-Way y Deafult-QoS. Las funciones y métodos del Módulo de Comunicaciones han sido configuradas de acuerdo a estas características.

Se diferencian dos puntos muy claros en la implementación, el "registro" y la "comunicación". El registro es el procedimiento mediante el cual se llevan a cabo todas las funciones y comandos necesarios para realizar los vínculos de la red de IEEE 1451. Para realizar la parte de la "comunicación" hace falta haber realizado (es muy recomendable) correctamente el proceso de registro.

Para realizar el registro se utilizan las interfaces Registration y NetRegistration del Módulo de comunicaciones.

Para realizar las funciones de "comunicación" se utilizan las interfaces Comm y NetComm del Módulo de Comunicaciones.

Se ha comenzado a realizar la implementación desde el "descubrimiento" y "registro" de la red Zigbee con el estándar IEEE 1451.

Registro IEEE 1451

Para iniciar el registro, es estrictamente necesario que los elementos y dispositivos formen una red, es decir, tengan vínculos entre ellos. Para ello, se crea un dispositivo como coordinador y a continuación se realiza binding con el resto de los dispositivos con los que queremos crear la red.

Cada dispositivo NCAP y WTIM debe de registrar su capa "Transducer Service API" (es decir, los servicios que tiene dele estándar) con al menos un Módulo de Comunicaciones. Esto parece trivial, pero se pueden dar casos en los que un dispositivo sea muy potente y trabaje con varias tecnologías a la vez (Wi-Fi, Zigbee…) por lo que tendrá que tener tantos Módulos de Comunicaciones como tecnologías distintas tenga como mínimo. Nuestro Módulo de Comunicaciones es una nueva hoja de programa en c ("*1451.5*") dentro del *Workspace* de la aplicación ZASA introducida con la herramienta IAR Workbench. Esta nueva hoja, contiene todas las funciones del Módulo de Comunicaciones y el acceso a ella se hace desde el programa principal. La función se llama "*registerModule( )*", para cada Módulo de comunicaciones registrado se asigna un identificador moduleId para usarlo en las posteriores llamadas.

Problemas direcciones Zigbee

La función*registerDestination()* sólo la realizan los dispositivos NCAP. Es importante recordar que los NCAP tienen que estar configurados como coordinadores (Zigbee) de la red.

La función *registerDestination()* registra un nodo final de la red Zigbee con el identificador *moduleId*(un Módulo de Comunicaciones) para convertirlo en un WTIM. De esta forma, el WTIM es identificado mediante un identificador *destId* único para dicho *moduleId*. Este identificador, es almacenado junto con el *moduleId*

al que pertenece y con la dirección Zigbee que tiene en la red para que pueda ser utilizada. Está dirección Zigbee del nodo final la determina y la almacena el coordinador en el momento en el que incluye al nodo final Zigbee en su red. Para ello, se han creado estructuras en lenguaje C para usarlas en la aplicación ZASA con el fin de almacenar los WTIMs (*destiIds*) que tiene conectados un módulo de comunicaciones de un NCAP (*moduleId*) y la dirección Zigbee que tiene cada *destId*.

No obstante, no se ha podido realizar de esta forma porque la dirección Zigbee del nodo final que guarda y establece el coordinador, es almacenada en el microprocesador esclavo CC2480 y como se ha explicado anteriormente, tan sólo tenemos "acceso" al código del MSP430 que actúa como maestro. Al ser una aplicación sencilla y de fácil utilización, carecemos de una forma de acceder a la información que contiene el microprocesador esclavo.

Existe otra forma de conocer la dirección Zigbee que le ha asignado el coordinador al nodo final. Dicha dirección es enviada por el coordinador al nodo final y estela almacena. Se puede enviar la dirección desde el nodo final al coordinador, pero para ello habría que implementar código propietario para que el coordinador sepa que la información que le llega es la dirección del nodo final Zigbee y estaríamos por tanto "traicionando" la filosofía "Plug&Play" que ofrece IEEE 1451.

Todo este problema radica en el kit eZ430-RF2480 junto con la aplicación ZASA, debido a que se trata de un kit de muy fácil manejo y desarrollo, sus limitaciones son grandes debido a la arquitectura basada en un procesador ZigBee/802.15.4 (CC2480) y un microprocesador (MSP430). El procesador ZigBee/802.15.4 (CC2480) gestiona tanto la capa física como las capas MAC y de red, dejando al microprocesador encargado únicamente de la parte de aplicación. Este sistema es más simple, pero menos flexible, de cara al desarrollador puesto que éste sólo se debe ocupar de la aplicación, dejando que el procesador ZigBee/802.15.4 (CC2480) gestione el resto de capas.

La aplicación ZASA realiza la incorporación de nuevos elementos de la red Zigbee realizando *binding*. El coordinador es el encargado de llevar a cabo el proceso de introducir a un nodo como miembro de la red Zigbee.

El *binding* (o vínculo) es una conexión lógica entre un punto de acceso origen y uno o varios de destino. Sólo se puede realizar un vínculo *binding* entre puntos de acceso que compartan el mismo *cluster*. Ya que un dispositivo puede poseer varios puntos de acceso, también puede soportar varios vínculos. Para poder vincularse ambos dispositivos deben estar ejecutando la misma aplicación (ZASA en este caso). Puesto que todo es correcto, se realiza la vinculación entre el coordinador y el nodo final. A partir de ese momento, el nodo enviará una medida de temperatura y otra de voltaje hacia el coordinador cada 10 segundos.

Esta red creada configura al coordinador con un único *cluster* de entrada pero ninguno de salida. El coordinador actúa como sumidero de la información, es decir, nunca enviará datos a sus dispositivos "hijo", exceptuando el caso de los paquetes "ack" (confirmación) a nivel MAC y de aplicación. Por el contrario, los nodos finales (*end_point*) tienen un *cluster* de salida pero ninguno de entrada, es decir, sólo envían datos. El *cluster* por el que recibe los datos el coordinador es el mismo para todos los *end points*, y en la aplicación del MSP430 no se puede identificar quién es el transmisor de los datos. Por tanto, no se pueden recuperar las direcciones Zigbee de los dispositivos.

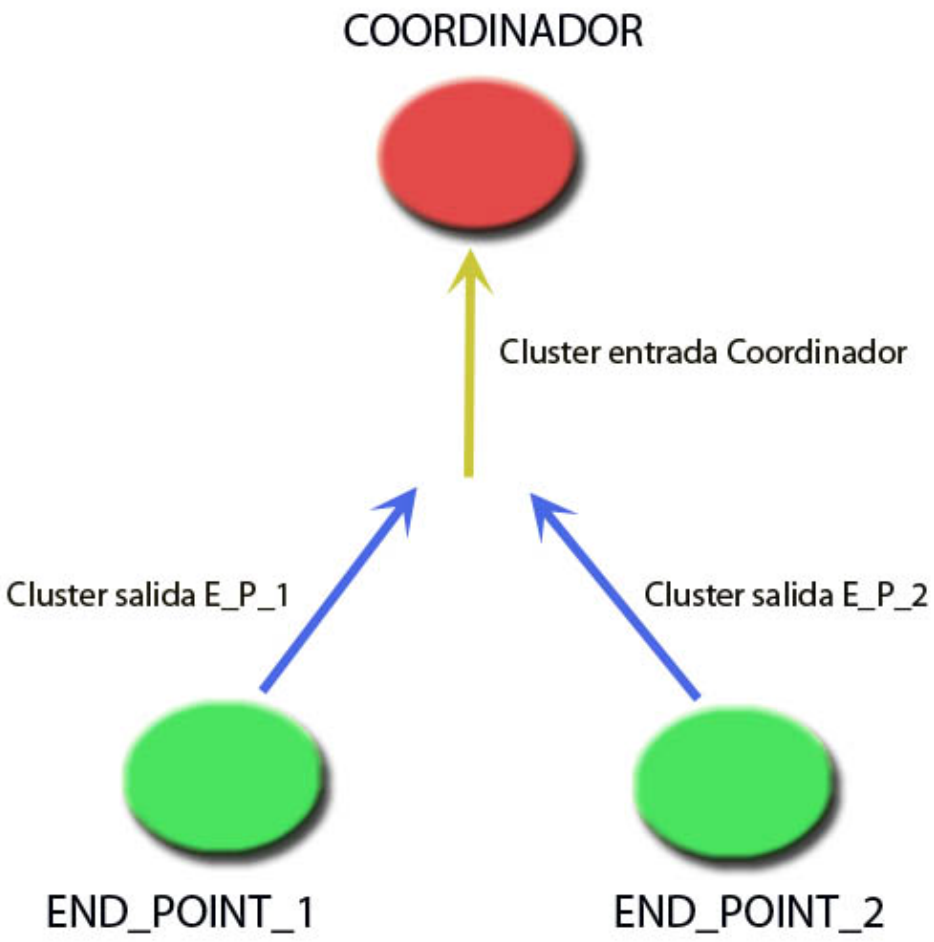


Ilustración 46: Cluster de Zigbee que componen la aplicación ZASA.

Esta forma de comunicación, es como la citada anteriormente *One-Way* que contempla IEEE 1451, y que en nuestro caso no sirve ya que habíamos adoptado la solución *Two-Way*.

La solución adoptada finalmente ha sido trabajar de la misma forma que lo hace la aplicación ZASA para vincular dispositivos, mediante *binding*.

Para poder distinguir a los *end points* Zigbee que componen la red, el *cluster* de salida de cada *end point* será diferente. Se va mantener los campos y atributos

que contiene el *cluster* pero el identificador del comando del transmisor ("SRCE_REPORT_ID" en la aplicación ZASA) será diferente en cada *cluster*. Este cambio obliga a que el *endpoint* tenga un comando distinto y además, el coordinador, tiene que tener tantos *clusters* de entrada como *end points* distintos quiera tener.

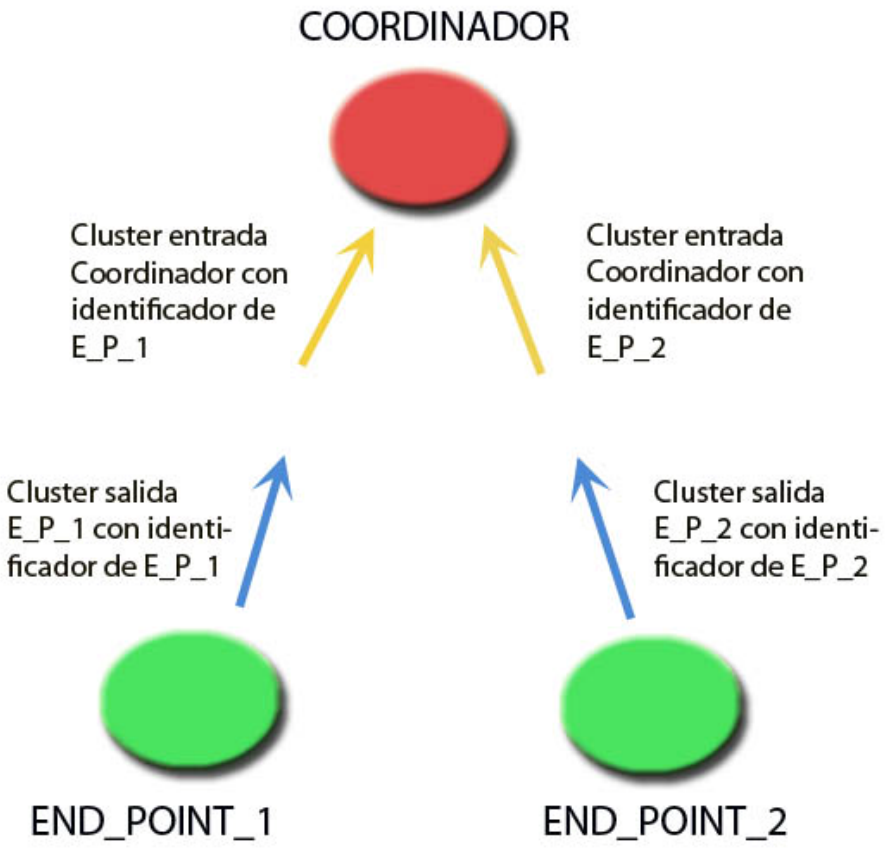


Ilustración 47:Nueva configuración de los Clusters Zigbee para llevar a cabo el registro de IEEE 1451.

Esta es la solución más parecida de realizar el proceso de "registro" de acuerdo con IEEE 1451. De esta forma, al realizar un *binding* distinto para cada nodo logramos almacenar fácilmente la dirección Zigbee que va a tener dicho nodo final junto con el comando (identificador del cluster). No obstante, cuando se realicen comunicaciones entre los dispositivos, debemos seguir utilizando el comando identificador de *cluster* en vez de la dirección Zigbee.

Se han creado estructuras para controlar el comando del *cluster*, la dirección Zigbee y el estado (variable booleana). De esta forma, controlamos y creamos una lista de los elementos que pertenecen a la red de IEEE 1451. Si algún WTIM se incorpora, o se borra lo controlamos con esta estructura.

Con el proceso relatado hasta el momento, se ha conseguido hacer el "registro" de cada miembro de la red Zigbee (coordinador y nodos finales) en los elementos del estándar IEEE 1451 (NCAP y WTIMs).

Se ha realizado un programa en la aplicación para observar los resultados de las estructuras que identifican a los miembros de la red. Para ello, utilizaremos la aplicación *Hyperterminal* para comunicarnos con el coordinador durante la ejecución en tiempo real. Utilizaremos la UART del MSP 430 para comunicarnos con el *Hyperterminal*. Enviaremos peticiones de las direcciones desde el *Hyperterminal* llamando a la función *halUARTRead*. Esta petición es un "*.txt"* que tiene la estructura de los mensajes de IEEE 1451. Al leer el coordinador el mensaje que le llega, lo procesa, coge la información de las estructuras que anteriormente hemos descrito (las direcciones Zigbee de los dispositivos que tiene vinculados) y lo devuelve al *Hyperterminal* a través de la función *halUARTWrite.* Pudiendo visualizar en la pantalla del *Hyperterminal* las direcciones Zigbee de los dispositivos WTIM que forman la "red" IEEE 1451.

Parte del código empleado ha sido creado para desarrollar la estructura de los mensajes y los comandos del estándar.

A continuación, una vez que se ha llevado a cabo el proceso de "registro", hay que crear el *cluster* Zigbee especificado por IEEE 1451.5 para realizar las comunicaciones de acuerdo al estándar.

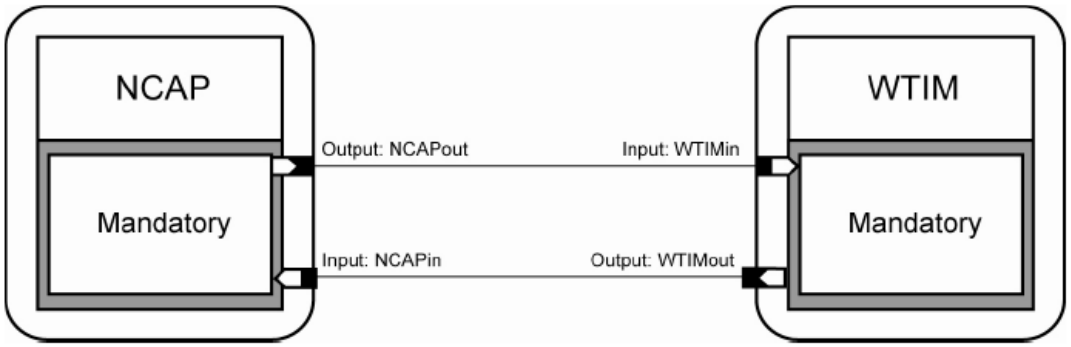


Ilustración 48: Cluster obligatorios en Zigbee para IEEE 1451.5.

Clusters Obligatorios		
Nombre Cluster	Cluster ID	Descripción
NCAPout	0x01	Cluster NCAP como emisor
NCAPin	0x02	Cluster NCAP como receptor
WTIMin	0x01	Cluster WTIM como receptor
WTIMout	0x02	Cluster WTIM como emisor

Tabla 42: Identificadores de los Clusters obligatorios en Zigbee 1451.5.

Por defecto, la aplicación ZASA permite comunicación en un único sentido, desde el*end point* hasta el coordinador. Hay que modificar la aplicación para conseguir una comunicación bidireccional y además dicha comunicación se tienen que hacer a través del *cluster* que especifica IEEE 1451.5 (ya visto anteriormente).

10. CONCLUSIONES Y LÍNEAS FUTURAS DE TRABAJO

En el presente capítulo expondremos de forma breve las conclusiones a las que hemos llegado tras la realización de este proyecto. Además, plantearemos una serie de posibilidades en cuanto a la ampliación del mismo.

10.1.CONCLUSIONES

Tras haber estudiado minuciosamente el estándar IEEE 1451 y

- Estándar a 3 niveles: IEEE 1451 ofrece la posibilidad de estandarizar una red de sensores desde el sensor hasta la comunicación con un usuario remoto. Se puede diferenciar 3 niveles (o partes) en los que se puede dividir el proceso de la estandarización de la red de sensores de acuerdo a IEEE 1451:
  - Nivel 1- Sensores: independientemente del fabricante del sensor o de la magnitud que mida, se puede implementar cualquiera de estos elementos como dispositivos del estándar, para ello, tan sólo se debe de implementar las TransducerChannel TEDS y/o Calibration TEDS (en función del sensor puede hacer falta alguna otra TEDS).
  - Nivel 2- Capa transporte: este estándar también ofrece independencia en el uso de la tecnología para comunicar un TIM con NCAP. Es decir, un NCAP puede operar con distintos TIMs a través de distintas tecnologías (bluetooth, Zigbee, CAN…)
  - Nivel 3- Red usuario: la comunicación desde el dispositivo NCAP hacia fuera, es decir, hacia un cliente remoto o servidor, también queda claramente definida usando *Restful* (Http) o STWS. Este es el punto más crítico para un NCAP que obliga a que se trate de un dispositivo más potente.

- Comandos y APIs: los comandos son mensajes destinados a cumplir una tarea o función y para llevarla a cabo, hacen uso de las funciones que contienen las APIs (ya sea Transducer Service o Module Communications). Es importante darse cuenta a la hora de la implementación que si no se prevé utilizar una serie de comandos habrá una serie de funciones y métodos en las APIs que no se llamarán y por tanto no hará falta implementarlas.

- Implementación en eZ430-RF2480: la implementación en el kit ez430-RF2480 no ha podido ser realizada al completo debido a los siguientes problemas:
  - Arquitecura *hardware*: este kit tiene un procesador ZigBee/802.15.4 que gestiona tanto la capa física como las capas MAC y de red, dejando al microprocesador encargado únicamente de la parte de aplicación. Al realizar la implementación sólo se ha podido ocupar de la aplicación gestionando el procesador ZigBee/802.15.4 el resto de capas sin poder gestionarlas nosotros.
  - Memoria RAM: la aplicación ZASA que viene cargada en el microprocesador 430 ocupa 800 bytes de memoria RAM de 1K que tiene disponible. En 200 bytes se ha podido llevar a cabo el proceso del "registro" y trabajar con los clusters.

- IEEE 1451.0 y IEEE 1451.5: a día de hoy y sin niguna duda, el miembro más importante de IEEE 1451 es IEEE 1451.0. es el presente del estándar y el cual define perfectamente la arquitectura interna de los dispositivos NCAP y TIM así como sus funcionalidades. Es el que permite la interoperabilidad entre distintos miembros del estándar. Por su parte, para la finalidad de este proyecto es también muy importante el miembro IEEE 1451.5, que es el encargado delas comunicaciones inalámbricas. El fututo de IEEE 1451 pasa por la comunicación inlámbirca NCAP-WTIM.

- Conocer IEEE 1451 y saber qué es lo que se necesita: la implementación de un estándar de la magnitud de IEEE 1451 es compleja y laboriosa. El primer paso es saber cuales son tus necesidades y que funcionalidades, posteriormente saber si IEEE 1451 se ajustan a ellas y cuales son sus implicaciones:
  - Tanto la interoperabilidad y como la versatilidad son características muy importantes e ineteresantes para una red de sensores, no obstante, no es recomendable adquirir interoperabilidad y versatilidad más allá de la necesaria para los requerimentos de la aplicación puesto que se puede traducir en la necesidad de dispositivos más potentes, caros e implementaciones más costosas, y para sistemas de bajo consumo no es interesante. No es necesario tener un NCAP con módulos de comunicaciones para Zigbee, Bluetooth y 6LoWPAN cuando inicialmente tansólo se piense trabajar con Zigbee (por ejemplo).

- Memoria: una aplicación sencilla de este estándar supone, en término medio, 500kbytes de memoria RAM y 3000 bytes de ROM usando lenguaje C.
  - Memoria RAM: la mitad de la memoria RAM la ocupa las TEDS. Hay que tener en cuenta que si un NCAP controla "n" sensores, son "n" TransducerChannels TEDS (cada una de ellas ocupa en torno a 100bytes) las necesarias, por lo que la memoria aumenta.
  - Memoria ROM: la mayoría de es ta memoria es debida a las funciones y métodos de las APIs. En un NCAP son bastante más los servicios que ofrecen respecto a un TIM, por tanto, también ocupará más.

10.2. LÍNEAS FUTURAS DE TRABAJO

Aunque el estándar IEEE 1451 tiene unos quince años de vida, sus miembros IEEE 1451.0 y IEEE 1451.5 apenas llegan a los dos años desde que fueron publicadas sendas normas. A continuación mencionaremos algunas opciones que, por su relación con el presente proyecto resultan muy interesantes para el futuro de IEEE 1451:

- Implementación: en la actualidad no existen prácticamente implementaciones del estándar para redes basadas en Zigbee y 6LoWPAN, se propone a partir de los conociemientos adoptados por este proyecto seguir con una implementación completa del estándar, para ello el dispositivo debe tener la siguiente arquitectura *hardware*:
  - Arquitectura basada en un transceptor 802.15.4 y un microprocesador. En este tipo de dispositivos el transceptor se ocupa de la parte radio (capa física 802.15.4) mientras que el microprocesador se encarga de gestionar la capa MAC, de red y de aplicación.
  - Arquitectura basada en un dispositivo completamente integrado (System-on-chip o SoC). En este último caso, tanto el transceptor como el microprocesador están integrados en un único chip.

  Para ello se propone el uso del microprocesador JN5148 de Jennic, que ofrece unas mejoras características que los JN5139, JN5121 aunque estos también son suficientes para llevar acabo la implementación. Cualquiera de estos dispositivos que tengan una arquitectura la descrita, son válidos.

  La versión ZigBee de Freescale es conocida como BeeStack. Respecto a la capa MAC, Freescale ofrece la posibilidad de usar el protocolo SMAC (simple MAC), una versión propia y simplificada de la capa MAC 802.15.4 que podría ser suficente.

- Uso LabView: en las implementaciones de de otros miembros del estándar de IEEE 1451 en las que sa ha usado un dispositivo y un PC para llevarla a

cabo, se ha usado en varias ocasiones el software LabView (para PC). Este *software* permite una fácil adquisición de datos en dispositivos periféricos del PC y además posee unas librerías para IEEE 1451 que se acaban de desarrollar para la programción del estándar (contiene TEDS, comandos…)

- Hub domiciliario, ISO/IEEE 11073-10471: sería muy interesante realizar una comparativa entre IEEE 1451 y la especialización "Hub de vida independiente- 10471" de ISO/IEEE 11073. Esta especialización de ISO/IEEE 11073 es el máximo acercamiento entre los dos estándares.

11. REFERENCIAS

A continuación se presentan los diferentes documentos usados para realizar este documento. Prácticamente la totalidad de :

"[1451.0] IEEE Std 1451.0-2007 IEEE"

Standard for a Smart Transducer Interface for Sensors and Actuators—Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats.

"[1451.2] IEEE Std 1451.2™-1997"

IEEE Standard for a Smart Transducer Interface for Sensors and Actuators—Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats.

"[1451.3] IEEE Std 1451.3™-2003"

IEEE Standard for a Smart Transducer Interface for Sensors and Actuators—Digital Communication and Transducer Electronic Data Sheet (TEDS) Formats for Distributed Multidrop Systems.

"[1451.4] IEEE Std 1451.4™-2004"

IEEE Standard for a Smart Transducer Interface for Sensors and Actuators— Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats.

"[1451.5] IEEE Std 1451.5-2007"

IEEE Standard for a Smart Transducer Interface for Sensors and Actuators—Wireless Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats.

"**A ZigBee wireless sensor network compliant with the IEEE1451 standard**".
Jorge Higuera, Jose Polo, Manel Gasulla, Instrumentation, Sensors and Interfaces Group, Universitat Politécnica de Catalunya, Castelldefels (Barcelona), Spain

"A study on the Implementation of IEEE1451 for e-Health"

Jaehwan Kim, Wooshick Kim, 1 Department of Information and Communication Engineering, Sejong University98 Kunja-Dong, Kwangjin Gu, Seoul, 143-747, The Republic of Korea

"A Synopsis of the IEEE P1451- Standards for Smart Transducer Communication"
Kang Lee National Institute of Standards and Technology 100 Bureau Drive, Stop 8220 Gaithersburg, Maryland 20899-8220

"STWS: A Unified Web Service forIEEE 1451 Smart Transducers"
Eugene Y. Song, Member, IEEE, and Kang B. Lee, Fellow, IEEE

"Next Generation Application ProcessorBased on the IEEE 1451.1 Standard and Web Services"
Vitor Viegas" 2, J.M. Dias Pereira' 2, P. Silva Girao2
'ESTSetuibal-LabIM, Instituto Politecnico de Setubal, 2910-761, SetIbal, Portuga

"Embedded systems with IEEE 1451.1 on internet"

Miroslav Sveda, Faculty of hitnrmation Technology, Brno Lniversity of Tecinology, 61266 Brno. Bozetechova 2, CZECH REPUBLIC

"Implementing IEEE 1451.1in a Wireless Environment"
Rick Schneeman, Computer Scientist. US Department of Commerce
National Institute of Standards and Technology (NIST)

"Sensor Alert Web Service for IEEE 1451-Based Sensor Networks"
Kang B. Lee National Institute of Standards and Technology 100 Bureau Drive, MS# 8220
Eugene Y. Song National Institute of Standards and Technology

"Service-oriented Sensor Data Interoperability for IEEE 1451 Smart Transducers"
Eugene Y. Song ,National Institute of Standards and Technology 100 Bureau Drive, MS# 8220Gaithersburg, Maryland USA 20899-8220, Kang B. Lee National Institute of Standards and Technology100 Bureau Drive, MS# 8220

"IEEE 1451: A Standard in Support of Smart Transducer Networking"
Kang Lee National Institute of Standards and Technology 100 Bureau Drive Stop 8220 Gaithersburg, Maryland 20899-8220 USA

"IEEE 1451.0 HTTP Command–response specifications"
10/2/08 Proposed by NIST

"An Implementation of the Proposed IEEE 1451.0 and 1451.5 Standards"
Eugene Y. Song Manufacturing Metrology Division National Institute of Standards and Technology100 Bureau Drive, MS# 8220 Gaithersburg, Maryland USA 20899-8220. Kang Lee Manufacturing Metrology Division National Institute of Standards and Technology 100 Bureau Drive, MS# 8220

"Understanding IEEE1451—NetworkedSmart TransducerInterface Standard"
Eugene Y. Song and Kang Le

"Unification of IEEE 1451 Family - Common Commands, TEDS &Functionality"

James Wiczer, PhD. Sensor Synergy, Inc. Buffalo Grove, IL[*www.sensorsynergy.com*](http://www.sensorsynergy.com)

"IEEE 1451.0 Compatible TEDS Creation Using .NETFramework"
Sindhu Manda, Deniz Gurkan Engineering Technology Department University of Houston, 394 Technology Bldg. T2, Houston, TX 77204-4021 USA

"Redes Inalámbricas de Sensores Inteligentes. Aplicación a la Monitorización de Variables Fisiológicas"
Héctor Ramos Morillo, Francisco Maciá Pérez, Diego Marcos Jorquera. Departamento de Tecnología Informática y Computación, Universidad de Alicante,

"IEEE 1451 Smart Transducer Standard for Sensor Interoperability"
Workshop on"From Sensors to Applications:Advancing the Interoperability of Ocean Sensors"Ocean IInnovattiion 2008Worlld Summiitt:: Ocean ObservattiionSt John's, CanadaOctober 22, 2008

"Implementing the IEEE 1451Network Capable ApplicationProcessor (NCAP) in aWindows Environment"
Jay Nemeth-JohannesSmart Sensor SystemsLoveland, Colorado

"Smart Transducer Web Services Based on theIEEE 1451.0 Standard"
Eugene Song, Kang LeeNational Institute of Standards and Technology100 Bureau Drive, MS# 8220Gaithersburg, Maryland USA 20899-8220

"IEEE 1451 –A universal transducer protocol standard"
Darold WobschallEsensors Inc.Amherst NY 14226 716-837-8719

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

PROYECTO FINAL CARRERA

ESTUDIO Y ELABORACIÓN DE UNA GUÍA DE IMPLEMENTACIÓN DEL ESTÁNDAR IEEE 1451 PARA REDES INALÁMBRICAS

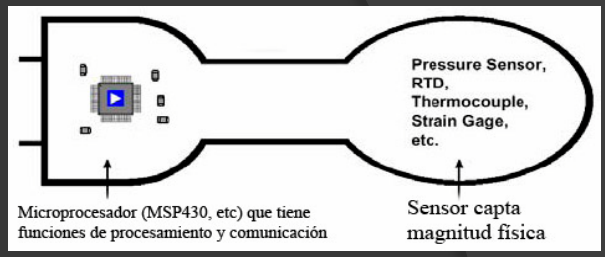
ÍNDICE

- Objetivos
- Introducción
- IEEE 1451
- Implementación
- Conclusiones

OBJETIVOS

- Estudiar y comprender el estándar para sensores inteligentes IEEE 1451
  - Visión objetiva y orientada a sensores inalámbricos
- Realizar una guía de implementación del estándar (centrada en sensores inalámbricos)
- Usar un Kit eZ430-RF2480 y comprobar cuales son las necesidades *hardware* del estándar

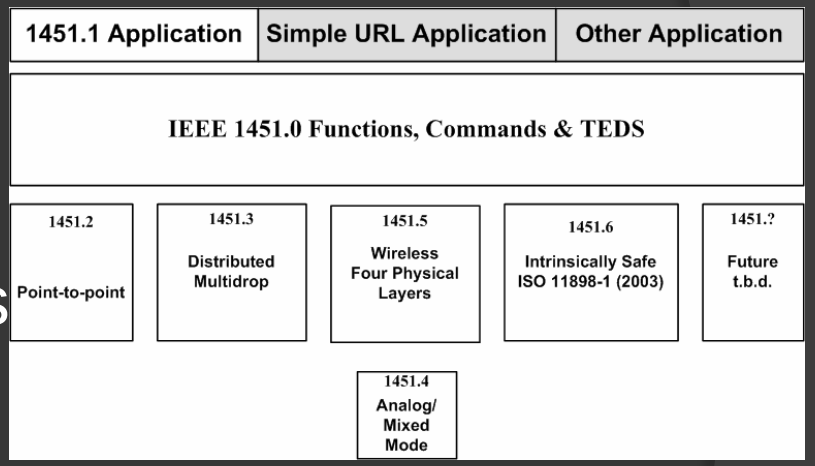
INTRODUCCIÓN



- Transductor → transductor inteligente
- Rápido desarrollo y aparición de sensores inteligentes y tecnologías
  - Creación de redes de transductores inteligentes es una solución muy económica y atractiva para una amplia gama de aplicaciones de medición y control
- Existen multitud de redes de sensores inteligentes cuyos datos pueden ser consultados a través de Internet
  - Cada red utiliza sus propios estándares, protocolos y formatos de representación de datos
- IEEE 1451

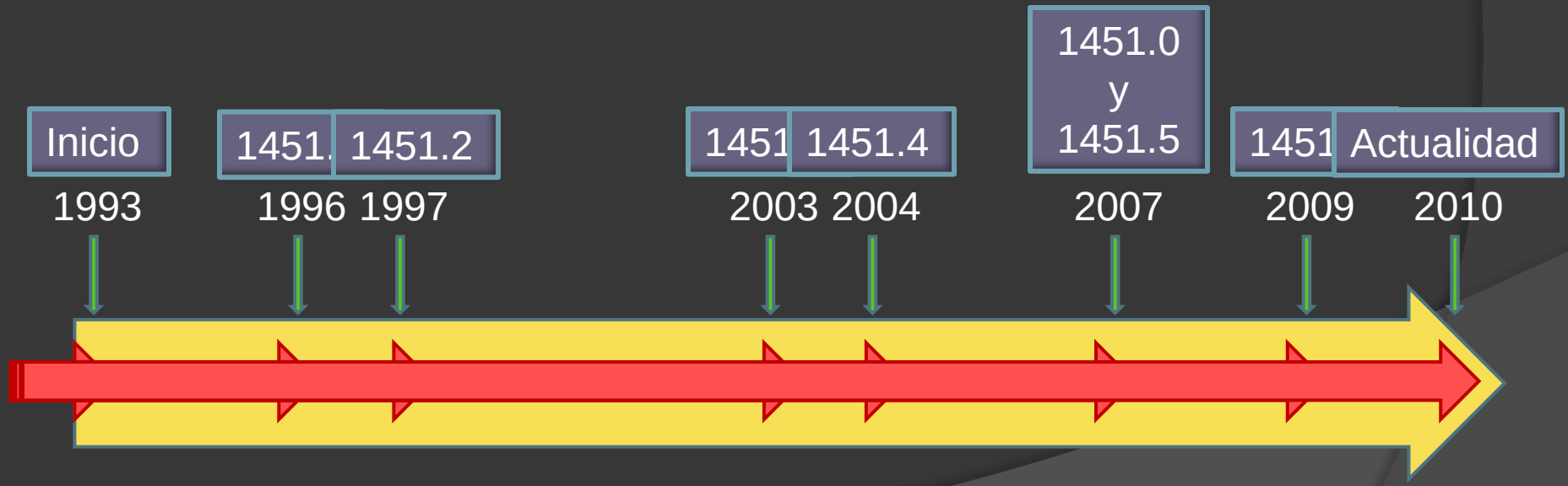
IEEE 1451

- Estándar redes sensores inteligentes
- Motivación
  - Interoperabilidad
  - Versatilidad
  - Compatibilizar tecnologías
  - *"Plug & Play"*
- Estándar ABIERTO
  - Accesible
  - Ampliación Fabricante
  - Revisión y nuevos miembros (IEEE y NIST)



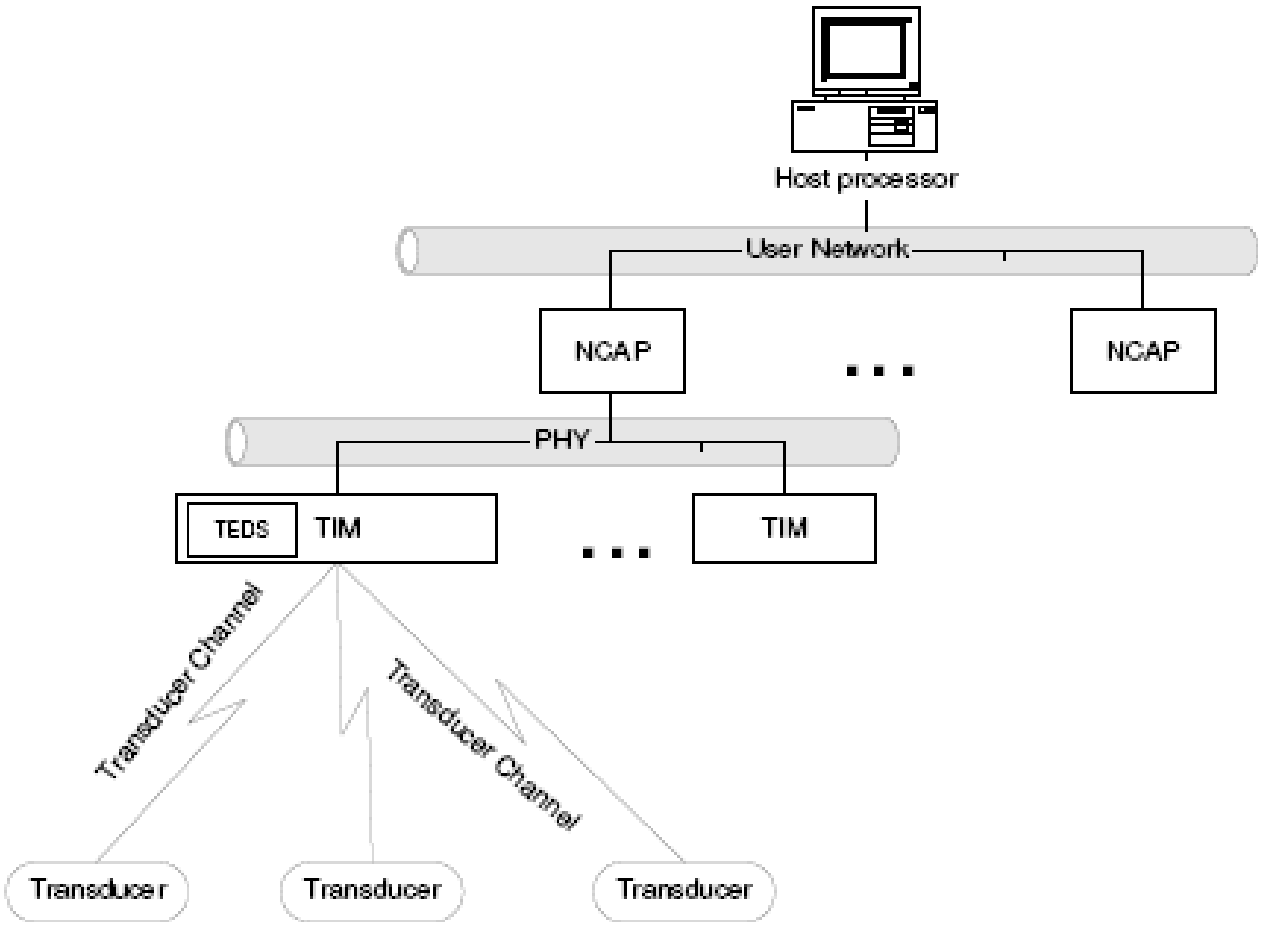
IEEE 1451

- Estándar formado por múltiples miembros
  - La importancia de cada uno ha variado
- Cronología

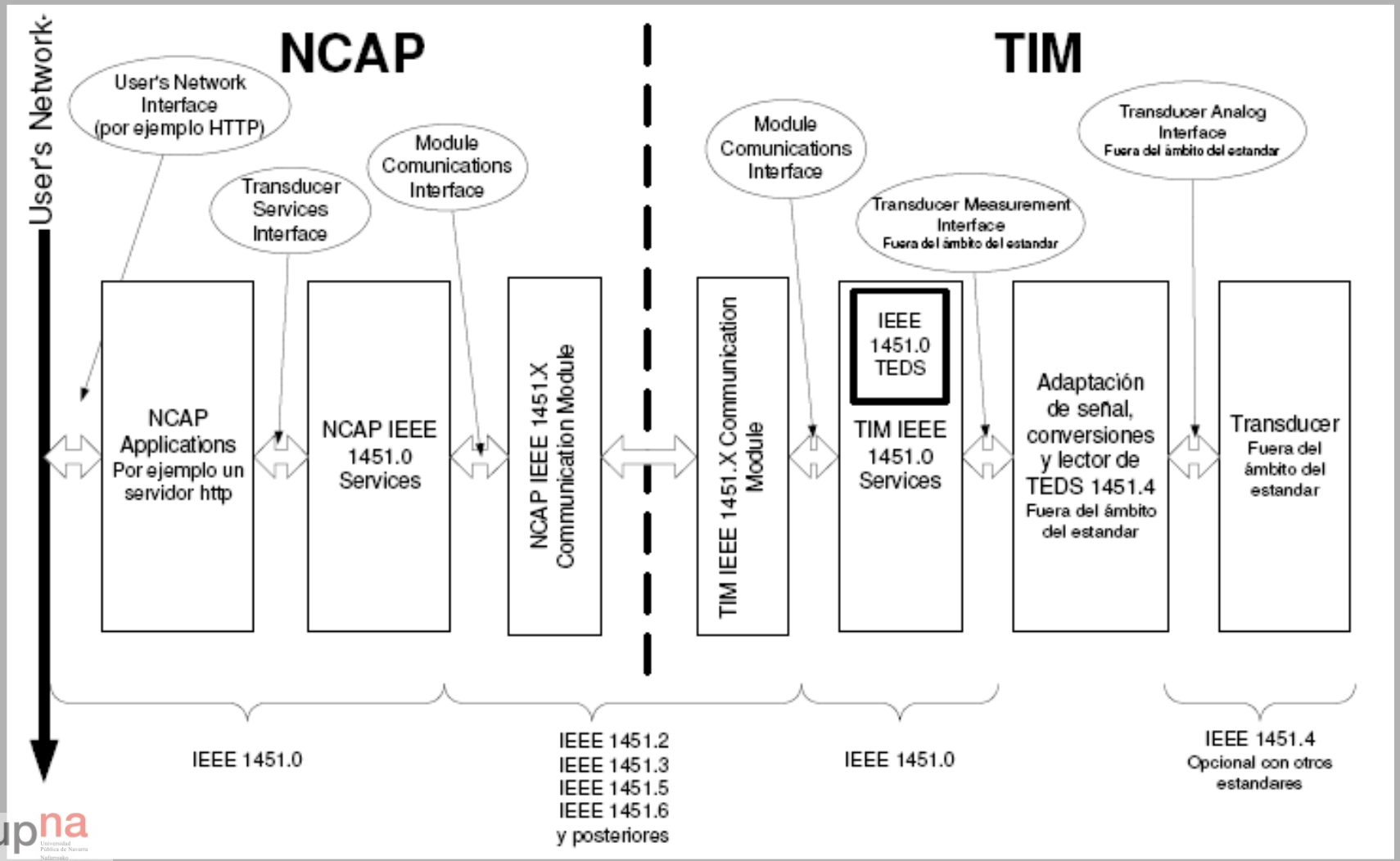


IEEE 1451

El estándar define los siguientes actores del sistema:.



IEEE 1451

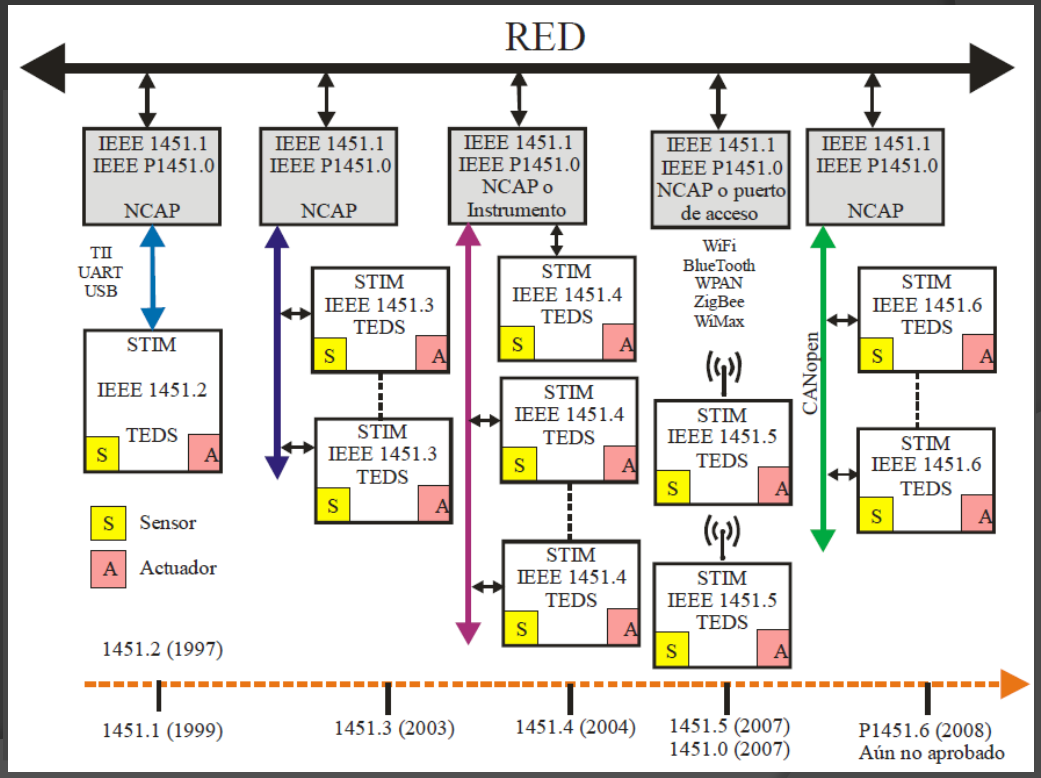


IEEE 1451

- NCAP
  - *Network Capable Application Processor*
  - Gestiona la información de su red
  - Se comunica con el exterior
- TIM
  - *Transducer Interface Module*
  - Comunica el transductor con el IEEE1451
- Transductor
  - Sensor, Event-Sensor, Actuador

IEEE 1451.0

- Miembro más importante (a día de hoy)
  - Interoperabilidad, a nivel de datos, entre los distintos miembros de IEEE 1451
  - Comandos, formatos TEDS y protocolos de comunicación (APIs)
- Muy necesario
  - Múltiples tecnologías
  - *"Estandariza los estándares"*



IEEE 1451.0

- Estructuras de los mensajes
  - Creados en la capa IEEE 1451.0
    - Inicia el NCAP
    - Respuesta
    - Inicia TIM

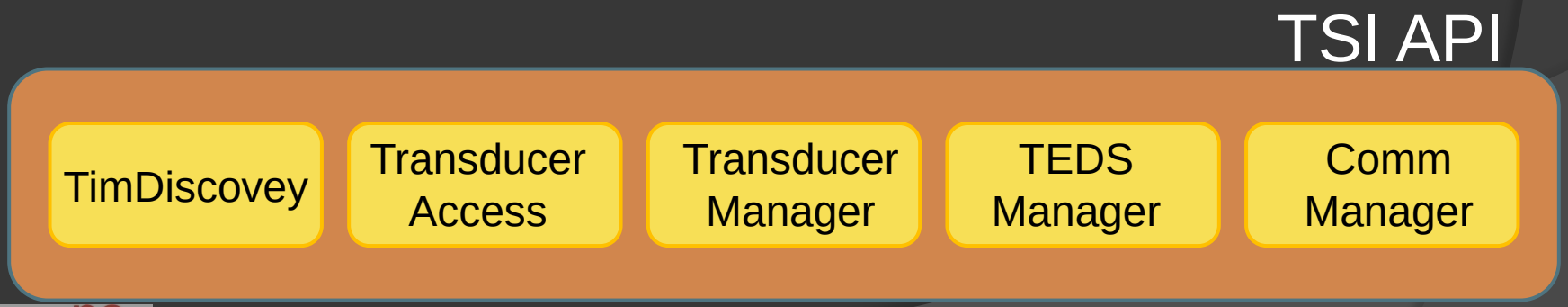
1-Octet							
7	6	5	4	3	2	1	0
Destination TransducerChannel Number (most significant octet)							
Destination TransducerChannel Number (least significant octet)							
Command Class							
Command Function							
Length (most significant octet)							
Length (least significant octet)							
Command dependent octets . . .							

1-Octet							
7	6	5	4	3	2	1	0
Success/Fail Flag							
Length (most significant octet)							
Length (least significant octet)							
Reply dependent octets . . .							

1-Octet							
7	6	5	4	3	2	1	0
Source TransducerChannel Number (most significant octet)							
Source TransducerChannel Number (least significant octet)							
Command Class							
Command Function							
Length (most significant octet)							
Length (least significant octet)							
Command dependent octets . . .							

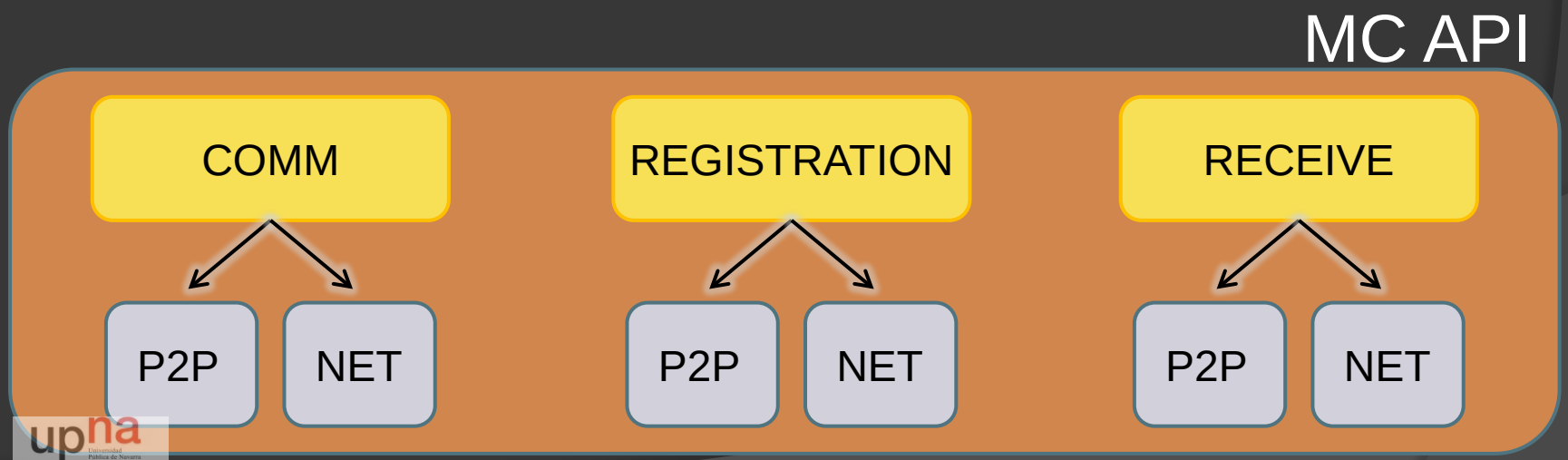
IEEE 1451.0

- Transducer Service API
  - Sólo en NCAP
  - Interfaz entre la aplicación (del NCAP) y las funciones de IEEE 1451.0
  - 6 Interfaces
    - 5 especificadas por el estándar

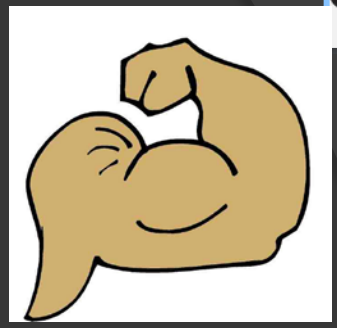


IEEE 1451.0

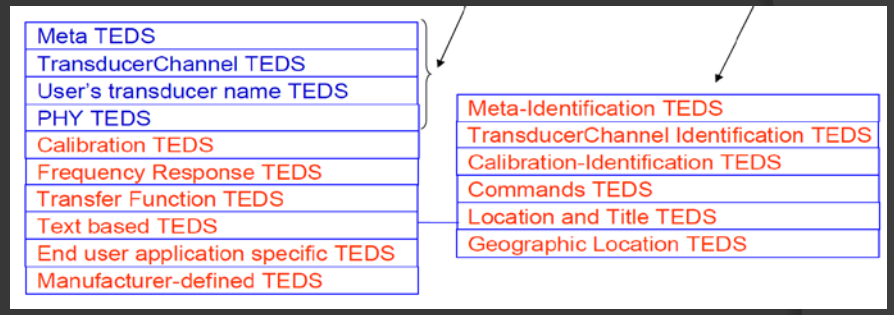
- Module Communications API
  - Responsable comunicación NCAP-TIM
    - Un Módulo de Comunicaciones por TIM
    - Múltiples Módulos de Comunicaciones por NCAP
  - Simétrico



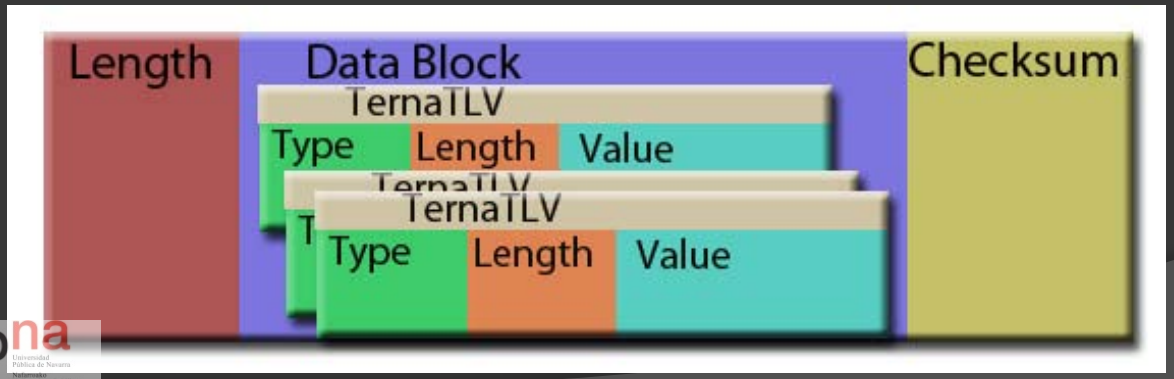
IEEE 1451.0



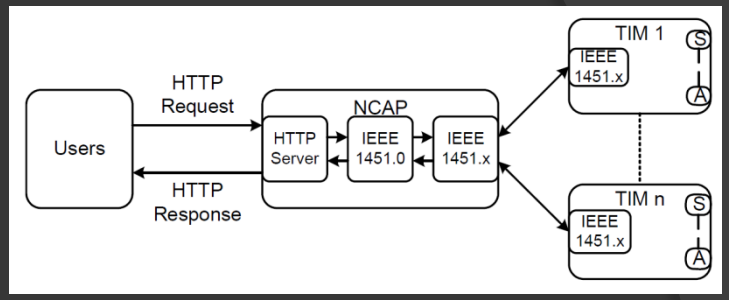
- TEDS
  - Método estandarizado de almacenar datos
  - Elemento clave
  - 4 TEDS obligatorias
    - Meta-TEDS
    - TransducerChannel TEDS
    - User´s Transducer Name TEDS
    - PHY TEDS → (IEEE 1451.X)



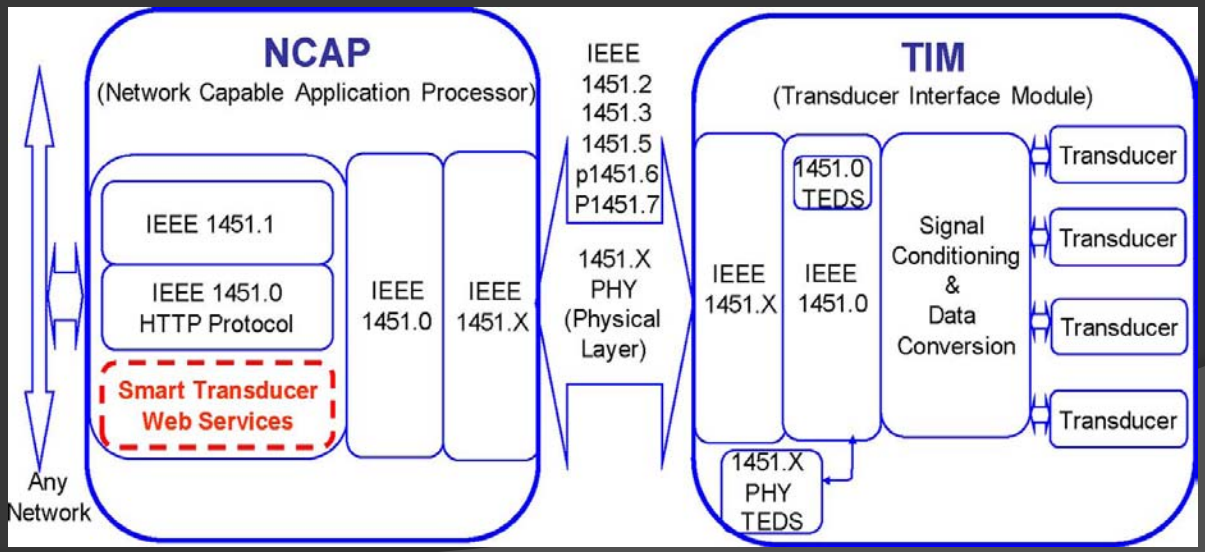
Field	Descripción	Tipo	Nº octetos
-	Longitud de TEDS	Uint32	4
1 a N	Bloque de Datos	Variable	Variable
-	Checksum	Uint16	2



IEEE 1451.0

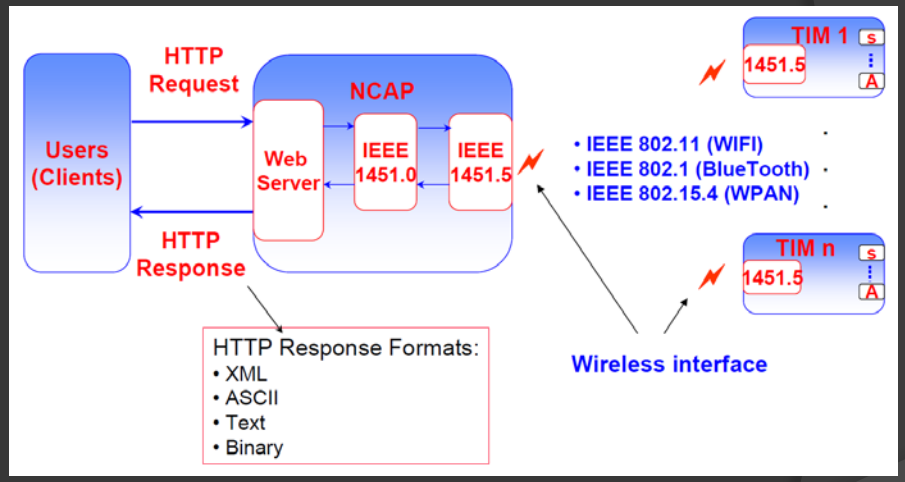


- HTTP. Nivel Aplicación
  - RestFul
    - http://<host>:<port>/<path>?<parameters>
      - Respuesta: XML, HTML o Text
  - Alternativas
    - IEEE 1451.1
    - STWS, (Smart Transducer Web Service)



IEEE 1451.5

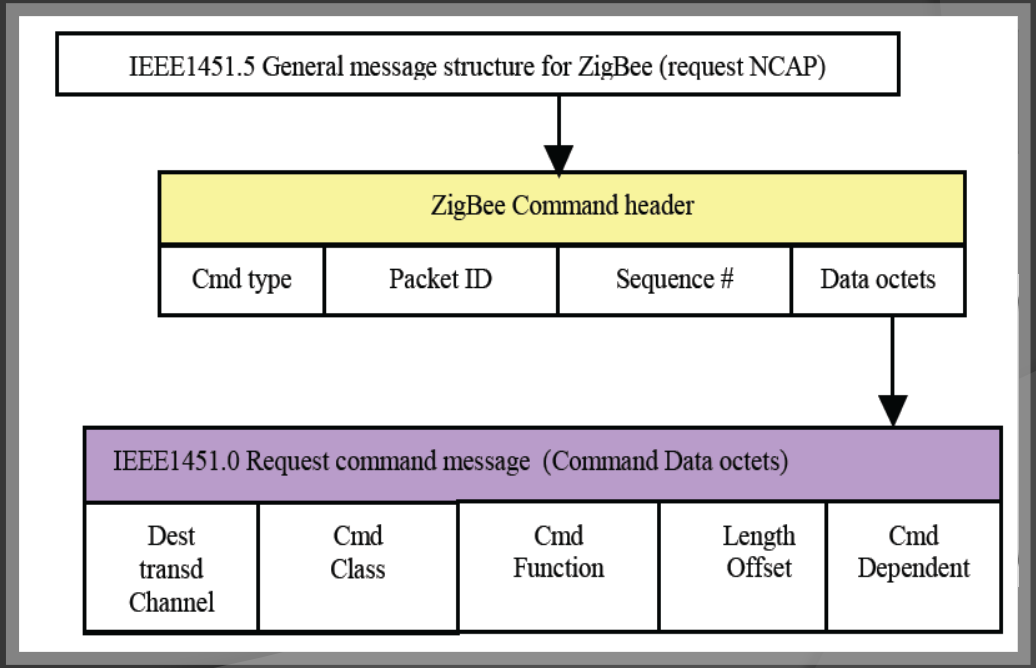
- Define capa transporte, MC
- 4 tecnologías
  - Wi-Fi, 802.11
  - Bluetooth
  - Zigbee
  - 6loWPAN
- PHY TEDS
  - Según tecnología



Smart Sensor Systems

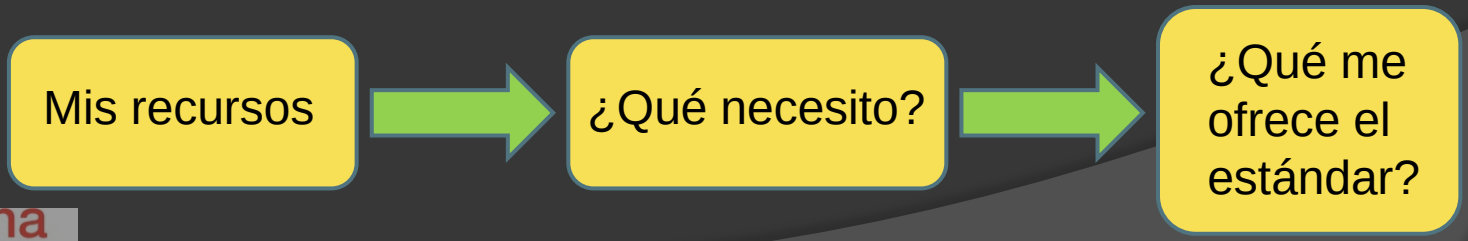
IEEE 1451.5. ZIGBEE

- Requisitos necesarios para proporcionar una capa de transporte para la aplicación de IEEE 1451.5
- Perfil de aplicación
  - ZigBee Private Profile Identifier: 0xBF00
- Clusters
  - 2 clusters definidos



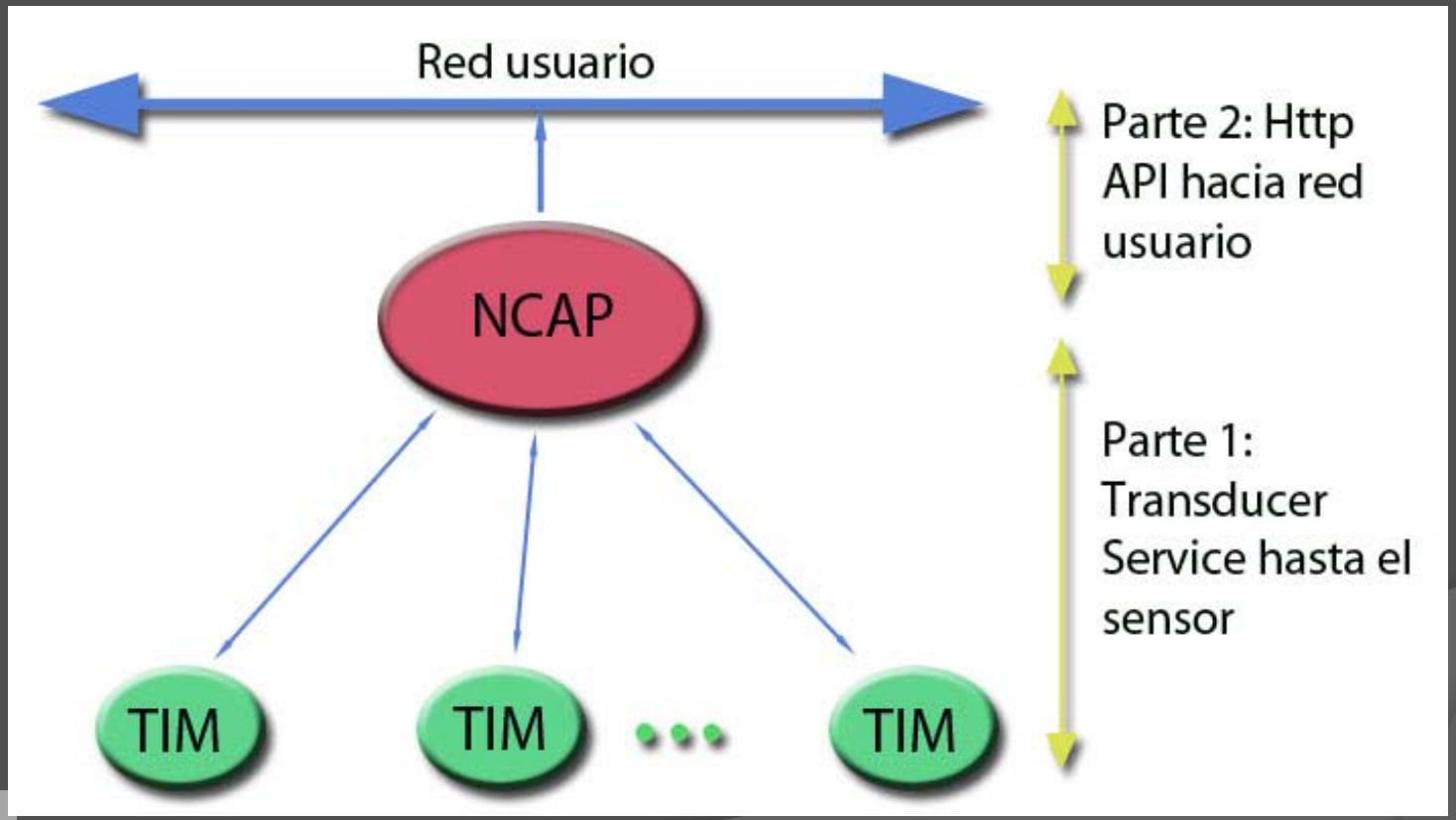
IMPLEMENTACIÓN

- IEEE 1451→ Estándar muy extenso
  - Necesidad
    - IEEE 1451.0
      - APIs, comandos, TEDS…
    - IEEE 1451.X → IEEE 1451.5
- Múltiples y muy variados Comandos
  - Previsión de los necesarios
    - Menos Comandos → Menos APIs



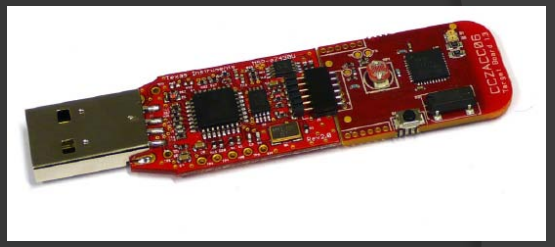
IMPLEMENTACIÓN

- 2 partes diferenciadas → Frontera TSI API



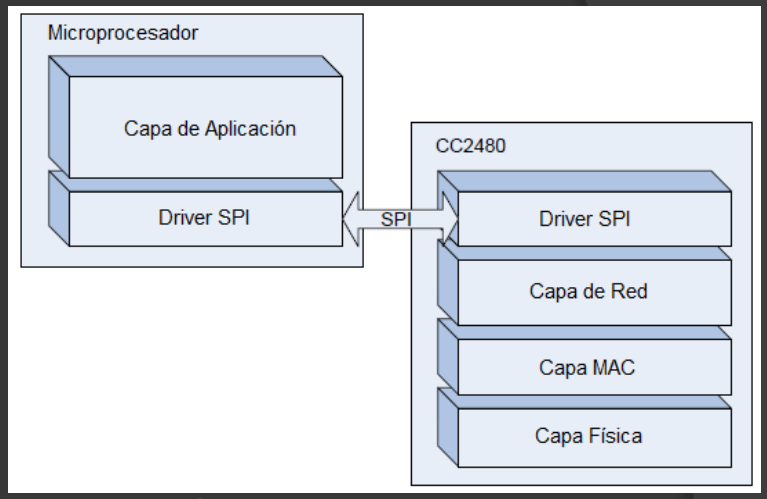
Kit eZ430-RF2480

- Kit eZ430-RF2480 *Texas Instruments*
- Arquitectura: procesador ZigBee /802.15.4 y Microprocesador
  - Procesador ZigBee/802.15.4, CC2480
    - Capa física
    - Capa MAC
    - Capa de red
  - Microprocesador 430 de TI encargado únicamente de la parte de aplicación.
- Más simplicidad pero menos flexibilidad

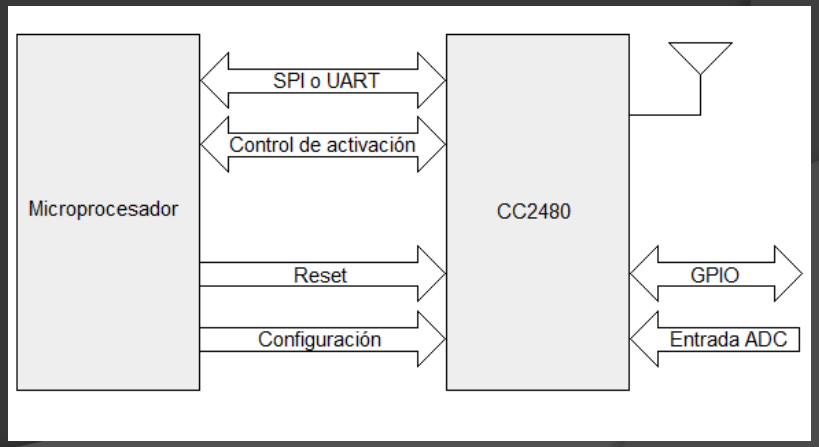


Kit eZ430-RF2480

- Control sobre MSP430
  - Características
    - 16 bits
    - 32KB Flash
    - 1KB RAM
  - Entorno *IAR Workbench*
    - Aplicación ZASA
    - Programación en C
    - Modificaciones
- Gestión de la red
  - CC2480



Distribución de la pila de protocolos entre microprocesador y CC2480



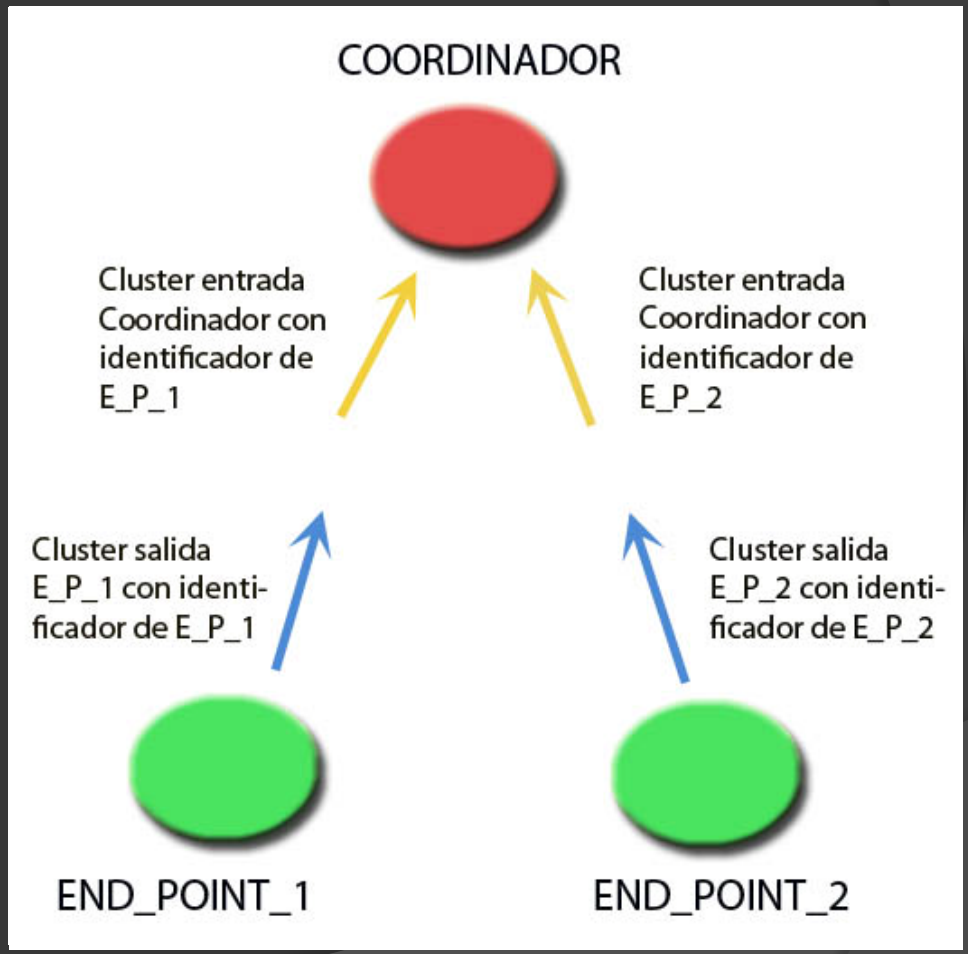
Conexión entre el integrado CC2480 y un microprocesador

Kit eZ430-RF2480

- Limitaciones
  - Limitación memoria RAM
    - MSP430F2272 → 1kB RAM
    - Aplicación ZASA ocupa en torno 800 bytes
  - Limitación gestión y control niveles de red
    - Debido a la arquitectura *hardware*
    - Direcciones Zigbee y MAC almacenadas en CC2480

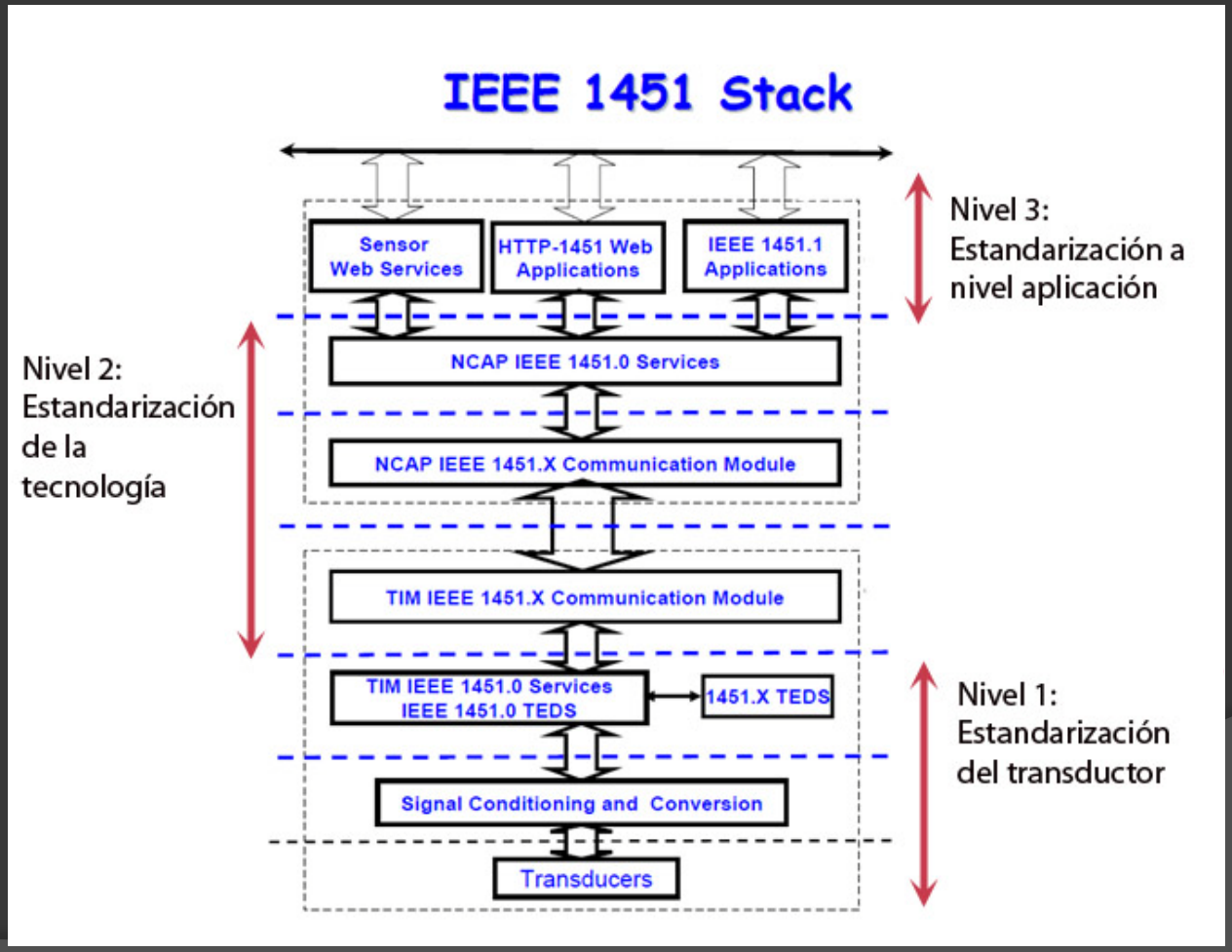
Kit eZ430-RF2480

- Modificación en aplicación ZASA
  - *Registration*
  - Binding



CONCLUSIONES

- Estándar a 3 niveles



CONCLUSIONES

- IEEE 1451.0 y IEEE 1451.5
- Requisitos *hardware*
  - Memoria: 500 bytes RAM y 3KB ROM
  - Arquitectura: Gestión y control de niveles de red y/o MAC
- Estándar extenso
  - Saber el grado de interoperabilidad necesario
    - + interoperabilidad + versatilidad → + coste

¡¡ MUCHAS GRACIAS POR SU ATENCIÓN !!

Ruegos y preguntas